Retour sur la méthodologie Machine Learning

Nicolas Baskiotis (nicolas.baskiotis@lip6.fr)
MLIA/LIP6, Sorbonne Université

Ce TP est destiné à revenir sur des concepts fondamentaux du Machine Learning à travers un exemple concret d'application : le problème de la recommandation. En très grande partie, les questions sont ouvertes vous invitant à explorer les différentes pistes évoquées.
Le problème de la recommandation a été rendu populaire par le célèbre challenge Netflix à la fin des années 2000. La plupart du temps, un tel problème implique un ensemble d'utilisateurs (*user*), un ensemble d'objets (*item*) et un ensemble d'informations sur ceux-ci (leur profil pour les utilisateurs, leurs caractéristiques pour les objets, les avis des utilisateurs sur une partie des objets) avec comme objectif de prédire les avis de nouveaux utilisateurs sur les objets (ou des utilisateurs sur de nouveaux objets). Les systèmes de recommandation sont utilisés dans de nombreuses applications, que ce soit sur les sites de e-commerce, les réseaux sociaux (nouveaux amis, articles d'intérêt), ... De nombreuses déclinaisons du problème existent, comme par exemple le "départ à froid" quand très peu d'informations est disponible sur le nouvel utilisateur, la détection de changement de dynamique dans les goûts des utilisateurs, les biais de notations. Nous étudierons dans la suite le problème dans un cadre usuel, sur des données issues du projet MovieLens [1] : il s'agit de données de recommandation sur des utilisateurs et des films. La section 1 est dédiée à la prise en main des données, la section 2 à l'étude de quelques algorithmes classiques de classification et à la mise en place des protocoles d'évaluation, la section 3 à l'étude d'algorithmes dit *instance based*, la section 4 aux techniques de factorisation matricielle.
Vous utiliserez en particulier les modules `sklearn, pandas, numpy, matplotlib.pyplot` dans la suite.

1 Prise en main des données et pré-traitement

Les données sont à télécharger sur `https://grouplens.org/datasets/movielens/`, nous travaillerons sur le fichier `ml-latest-small.zip` (700 utilisateurs, 9000 films). L'archive contient en particulier la liste des films `movies.csv` (leur identifiant, le titre, et les genres parmi 22 catégories) et `ratings.csv` la liste des notes attribuées par les utilisateurs.

Q 1.1 Calculer dans un premier temps :

- la matrice de vote avec en ligne les films et en colonnes les utilisateurs, chaque case indiquant le vote ;
- la matrice de descriptions des films : il faut binariser les genres, i.e. une colonne par genre ; un 1 indique la présence du genre pour le film, un 0 l'absence. Ajouter à cette matrice de description en dernière colonne le nombre de vote que a reçu ce film et le vote moyen obtenu. Vous pouvez également ajouter l'année de production du film (il est présent dans le titre du film).

Q 1.2 Explorer les données en regardant en particulier :

- la matrice des votes (vous pouvez l'afficher graphiquement)
- la distribution du nombre de votes des utilisateurs
- la distribution du nombre de vote par film
- la distribution de la moyenne des votes de chaque utilisateur
- la corrélation entre le nombre de votes pour un film et sa note moyenne

Q 1.3 Selon vos constatations, quelle(s) difficulté(s) peuvent se présenter pour les techniques de Machine Learning ?

1. `https://movielens.org/`

Q 1.4 Dans la suite, pour évaluer nos méthodes, nous utiliserons principalement les courbes ROC. Pour cela, nous avons besoin de binariser la décision et non pas d'attribuer un score à un film (j'aime ou j'aime pas). Quel seuil proposez-vous sur le score pour établir la limite entre les deux classes ? Est-il judicieux de faire un seuil par utilisateur ?

2 Classifieurs binaires pour la recommendation

Dans cette section, nous allons considérer uniquement les informations de genre sur les films, sans les informations de rating individuel (par utilisateur). L'objectif est de construire un classifieur unique pour tous les utilisateurs qui prédit en moyenne si le film est apprécié ou non.

Q 2.1 Proposer un protocole expérimental pour 1) traiter les données afin d'en faire un problème de classification binaire, 2) évaluer votre modèle avec une courbe ROC.

Q 2.2 Tester quelques classifieurs usuels : forêts aléatoires, boosting, k-plus proches voisins. Commenter vos résultats.

3 Instance based learning

L'approche précédente n'est bien sûr pas performante. Une première solution au problème de recommandation (individuel) est inspirée de l'approche k-plus proches voisins. L'idée est de considérer qu'un utilisateur aura tendance à aimer ou ne pas aimer les mêmes films que les utilisateurs proches de lui en termes de votes sur d'autres films. Il faut pour cela :

- définir une similarité entre deux utilisateurs en fonction des films qu'ils ont notés en commun
- pour un utilisateur et un film à prédire, agréger les votes des utilisateurs proches de lui et retourner une décision en fonction de cette agrégation.

Les votes peuvent être agréger soit de manière pondéré (par la similarité entre utilisateurs), soit de manière brute.

Q 3.1 Pour calculer la similarité entre utilisateurs, il est recommandé d'utiliser la distance cosine $d(\mathbf{x}, \mathbf{x}') = 1 - \frac{<\mathbf{x}.\mathbf{x}'>}{\|\mathbf{x}\|\|\mathbf{x}'\|}$ plutôt que la distance euclidienne (pourquoi ?). Calculer la matrice de similarité des utilisateurs avec les deux variantes et afficher là.

Q 3.2 Dans ce contexte, il n'est pas possible de séparer en ensemble de test et d'apprentissage une partie des films et une partie des utilisateurs, puisqu'une information sur les deux est nécessaire afin d'émettre une recommandation. Le découpage se fait sur les votes exprimés, une partie des votes est retirée de la matrice afin d'évaluer l'algorithme. Quelles précautions prendre lors de ce découpage ?

Q 3.3 Implémenter un algorithme de recommandation (avec ou sans vote pondéré) et tracer la courbe ROC associée.

Q 3.4 Calculer la matrice de similarité entre films cette fois et afficher la. Voyez vous un problème/biais apparaître sur certains films qui peuvent fausser les résultats (surtout sur les utilisateurs qui ont peu d'avis exprimés) ? Proposer une solution (en vous inspirant du traitement tf-idf sur les textes).

Q 3.5 Une autre manière de procéder est de s'intéresser au voisinage du film (et non pas au voisinage de l'utilisateur) : le vote prédit est l'agrégation des votes que l'utilisateur a donné aux films proches du film considéré en utilisant la matrice de similarité des films. Implémenter et comparer les deux approches.

Q 3.6 Quelles sont les limites de ces méthodes dites instance based ?

4 Factorisation matricielle

Soit un ensemble de m utilisateurs U et un ensemble de n items I, et une matrice R de dimension $m \times n$ telle que $r_{u,i}$ représente la note que l'utilisateur u a donné à l'item i, 0 si pas de note correspondante (on suppose les notes > 0). Dans cette partie, l'hypothèse est qu'il existe un espace latent de dimension d commun aux utilisateurs et aux items, qui permet d'expliquer par une combinaison linéaires les goûts des utilisateurs. Dans le cadre de la recommandation de films par exemple, cet espace pourrait décrire des genres de films. A chaque utilisateur u correspond un vecteur de pondération x de taille d qui indique les intérêts de l'utilisateur en fonction de chaque genre ; à chaque film i correspond un vecteur de pondération y de taille d qui indique la corrélation du film avec chaque genre.
L'objectif est ainsi de trouver deux matrices X et Y de tailles $m \times d$ et $d \times n$ telles que $R \approx XY$. On utilise en général les moindres carrés comme fonction de coût pour calculer l'erreur de reconstruction. La fonction à optimiser est $min_{X,Y} \sum_{u,i|r_{u,i}>0} (r_{u,i} - y'_{.,i} x_{u,.})^2 + \lambda(||y_{.,i}||^2 + ||x_{u,.}||^2)$. Une manière d'optimiser la fonction est par descente de gradient stochastique, avec les formules de mise-à-jour suivantes :

- $e_{u,i} = r_{ui} - y'_{.,i} x_{u,.}$
- $y_{.,i} = y_{.,i} + \gamma(e_{u,i} x_{u,.} - \lambda y_{.,i})$
- $x_{u,.} = x_{u,.} + \gamma(e_{u,i} y_{.,i} - \lambda x_{u,.})$

Cette formulation peut être améliorée en intégrant le biais des utilisateurs et des items (que l'on peut facilement introduire).
Une autre manière de procéder est de faire une décomposition SVD en 3 matrices telle que $R \approx U \Sigma V^T$, Σ étant une matrice carré de dimension l'espace latent. Que signifie alors les matrices U et V ?

Q 4.1 Choisir l'une des méthodes (à implémenter ou en utilisant sklearn) et tester.

Q 4.2 Afin de pouvoir visualiser les données, il est nécessaire d'avoir une réduction de dimension très forte (en 2 ou 3d ...). L'objectif de la plupart des algorithmes dans ce domaine est de préserver les distances locales lors des projections. Deux exemples d'algorithmes sont MultiDimensional scaling (MDS) et t-Stochastic Neighbor Embedding (t-SNE). MDS utilise une approche algébrique par identification de valeurs propres à partir d'une matrice de similarité entre données ; t-SNE utilise une modélisation probabiliste en étudiant la KL-divergence entre la distribution des points dans l'espace originale et dans l'espace projeté. Scikit-learn implémente ces deux algos. Etudier les résultats précédents en visualisant les données à partir de la nouvelle représentation et par MDS ou t-SNE. Vous pouvez utiliser pour cela une similarité cosinus pour MDS, et les coordonnées des films dans l'espace latent trouvé dans les questions précédentes pour t-SNE. Utiliser l'information sur les genres pour analyser vos résultats.