Script oral

Lila, Julie, Eva

1 - Introduction :

****

Diapo 2

Un système de recommandation est une application permettant de filtrer des informations afin de proposer à un utilisateur des éléments susceptibles de l'intéresser selon son profil.
Il existe 4 familles d'algorithmes de recommandation :

- Ceux basés sur la personnalisation
- sur le contenu
- le filtrage collaboratif
- Et les hybrides qui combinent plusieurs approches.

Ils sont utilisés dans plusieurs plateformes comme les réseaux sociaux et les sites de streaming. Nous nous pencherons sur le filtrage collaboratif.

Diapo 3

Dans un premier temps, nous décrirons les jeux de données utilisés. Puis nous expliquerons les 4 algorithmes implémentés :

- Le prédicteur de base
- Les plus proches voisins
- La descente de gradient stochastique
- Les moindres carrés alternés.

****

****

Diapo 4

Le premier jeu de données MovieLens a plus de films que d'utilisateurs et des notes allant de 0,5 à 5. Les autres jeux de données Jester Jokes, ont plus d'utilisateurs que de blagues et des notes allant de -10 à 10. Les jeux de données sont de plus en plus grands.
On note R la matrice contenant l'ensemble des notations des utilisateurs pour les items .
Cette matrice est en grande partie vide car tous les utilisateurs n'ont pas noté tous les items.

Diapo 5

Pour évaluer la performance de nos algorithmes, nous nous baserons sur deux métriques, la MAE et la RMSE. Elles mesurent la taille de l'erreur commise en comparant la note prédite à la note réelle. Plus le score est faible, meilleure est la prédiction. Pour calculer les scores, on supprime 20% des notes sur la matrice R que l'on va chercher à prédire avec nos algorithmes.

****

****

2 - Prédicteur de base :

Diapo 6

Le premier algorithme de recommandation que nous pouvons étudier est le prédicteur de base. En effet, il s'agit d'un des algorithmes les plus faciles à implémenter mais en dépit de la précision. C'est pour cette raison que cet algorithme est considéré comme le seuil de performance minimum pour les autres algorithmes. Les prédicteurs de base sont aussi utilisés comme algorithmes de secours lorsqu'un algorithme plus avancé échoue à établir une prédiction sous des conditions extrêmes.

Diapo 7

Dans cette partie, nous expliquerons les formules mathématiques utilisées par les prédicteurs de base pour calculer la prédiction. La prédiction consiste à calculer pour un utilisateur u et un item donné i, la note supposée que u aurait accordé à i.
Clic Le score se calcule de la manière suivante.

Clic mu où est la moyenne de l'ensemble des notes de notre base de données .

Clic bi est le biais de l'item i. Il indique si l'item a été plus ou moins bien noté que la moyenne générale de tous les items.

Clic bu est le biais de l'utilisateur. Il indique si l'utilisateur a tendance à donner des notes plus ou moins élevées que la somme de la moyenne et du biais des items. Par défaut, si un utilisateur ou un item n'ont pas de note, leur biais est nul.

Clic Enfin, on introduit aux calculs des biais un terme d'amortissement β permettant de contrebalancer les cas où il y a peu de notes pour un utilisateur ou un film.

Remarque : quand on code l'algorithme il faut introduire une correction car il arrive que la note prédite ne se situe pas dans l'intervalle des notes autorisées qui va par exemple de 0.5 à 5 pour Movielens. Alors, on force le résultat aux bornes de l'intervalle.

Diapo 8

Intéressons nous maintenant aux performances de notre prédicteur de base. Sur le jeu de données de Movielens, le MAE obtenu est de 0.68 et le score RMSE est de 0.89. Avec l'introduction d'un terme de damping β fixé à 20, les scores diminuent légèrement. Enfin, sur le jeu de données Jester le prédicteur de base obtient un MAE autour de 3.5 et un RMSE autour de 4.4. Avec l'augmentation de la taille des jeux de données jester, le RMSE décroît mais pas le MAE.

Clic En conclusion, l'avantage principal du prédicteur de base est qu'il est un algorithme simple à implémenter et robuste, qui fournit une prédiction dans la plupart des cas. (PAS DANS TOUS?)

****

****

3 - Algorithme des plus proches voisins

Diapo 9
Il existe deux types d'approches pour l'algorithme des plus proches voisins. Nous avons étudié, l'approche user-user qui consiste à trouver les utilisateurs similaires à l'utilisateur u pour lequel on veut faire une prédiction. Les utilisateurs voisins sont des utilisateurs qui ont acheté par le passé les mêmes items que l'utilisateur u et qui leur ont donné une note équivalente. D'autre part, l'approche item-item consiste à trouver des items voisins de celui pour lequel on veut faire une prédiction.

Diapo 10
La première étape consiste à calculer la similarité de l'utilisateur u à tous les autres utilisateurs du jeu de données. On ne prend en compte que les items notés par les deux utilisateurs. On reconnaît la formule du coefficient de corrélation.

Clic : Il faut cependant introduire ici deux corrections. Premièrement, on ne calculera la similarité entre deux individus que si le nombre d'items qu'ils ont tous les deux noté est supérieur à un certain seuil, autrement deux utilisateurs qui n'auront noté qu'un seul item en commun auront une corrélation égale à 1 alors qu'ils ne seront pas du tout similaires. En général, le seuil T=50 pour ne pas fausser les similarités ou exiger trop d'items communs. Deuxièmement, si des utilisateurs ont accordé la même note à tous les films qu'ils ont regardé, alors on a un dénominateur nul. Dans ce cas-là, on considère la similarité nulle, ce qui revient à supprimer l'utilisateur de notre jeu de données.
Clic : A l'issue de cette première étape, on obtient la matrice des similarités S entre tous les utilisateurs.

****

****
Diapo 11
Ensuite, le set des plus proches voisins peut être établi, il contient les utilisateurs dont les similarités avec l'utilisateur u considéré sont les plus élevées.

Clic : Et, parmi ces utilisateurs, on ne conserve que ceux qui ont noté l'item i et on forme le set Nui. Sa taille se situe généralement entre 20 et 60 utilisateurs, elle dépend du jeu de données.
Clic : On constate que les erreurs MAE et RMSE sont minimales lorsque le set est fixé à 23 utilisateurs pour Movielens,
Clic : et 58 utilisateurs pour Jester. L'algorithme des plus proches voisins semble plus adapté au deuxième jeu de données puisque globalement les scores décroissent quand la taille du set augmente. Or dans un jeu de données, plus on peut prendre en compte un nombre important de proches voisins, plus la prédiction est fiable car elle se base alors sur davantage de notations.

Diapo 12 Enfin on calcule le score. C'est une moyenne pondérée par les similitudes des utilisateurs avec l'utilisateur u considéré.

Clic : Cependant, il arrive que l'algorithme des plus proches voisins échoue. Par exemple, quand on construit le set des plus proches voisins, il se peut que ce set soit vide. Si l'utilisateur a noté très peu de films alors le set Nu sera très réduit et si parmi ses voisins aucun n'a noté l'item considéré alors le set Nui sera vide. Dans ces cas là nous avons choisis de faire appel au prédicteur de base pour calculer le score, c'est donc une hybridation.

Diapo 13
Sur le jeu de données MovieLens, le score MAE est de 0.66 et le score RMSE est de 0.88.
Clic : La prédiction est donc meilleure que celle du prédicteur de base.
Clic : Cela dit, les scores sont assez proches, cela peut s'expliquer par les failles de l'algorithme, comme l'arrivée de nouveaux utilisateurs dans le jeu de données. A l'inverse, le prédicteur de base n'est pas sensible à ces perturbations et fournit une prédiction dans la plupart des cas.
Sur le jeu de données Jester, on obtient un MAE de 3.39 et un RMSE de 4.46.
Clic : L'amélioration par rapport au seuil du baseline est meilleure que précédemment, car le jeu de données contient beaucoup plus d'utilisateurs que d'items.

****

****

4 - Descende de Gradient Stochastique ou Funk-SVD

Diapo 14
Un des principaux défauts de l'algorithme des plus proches voisins est qu'il détecte les relations par paires entre utilisateurs et items sans tenir compte de structures plus larges dans les données.
Par exemple, si les items représentent des films, nous pouvons imaginer qu'il existe une relation entre la note des utilisateurs et le contenu des films.
Les techniques de factorisation matricielle seraient donc plus efficaces car elles permettraient de découvrir ces caractéristiques latentes (ou cachées) qui déterminent comment un utilisateur évalue un item, permettant de faire des recommandations.

Diapo 15
Clic1
On rappelle R la matrice des notations et on introduit K le nombre de caractéristiques latentes à déterminer.

Clic2
Le but est alors de calculer les matrices P et Q telles que le produit entre P et Q transpose, noté R-tilde, approxime au mieux R. R-tilde est la matrice des notations prédites où aucune entrée n'est vide.

Clic3
Les coefficients de R-tilde se calculent donc de la façon suivante, rui-tilde représentant la note prédite de l'utilisateur u pour l'item i .

Clic4
L'algorithme va entraîner et afin de minimiser les différences entre les notes réelles rui et les notes prédites rui-tilde.

****
****

Diapo 16
Clic1
Pour minimiser cette différence, nous utilisons l'algorithme de descente de gradient stochastique. Cette méthode consiste à trouver itérativement le lieu du minimum d'une fonction différentiable en partant d'un point initial. Alpha est ici le taux d'apprentissage.

Clic2
La descente de gradient connaît un intérêt en machine Learning.
Considérons N+1 données où les vecteurs Xi représentent les entrées du système et les scalaires yi la sortie attendue. Nous souhaitons trouver un modèle mathématique F caractérisé par des paramètres (a1...an) qui approxime au mieux les yi .
La descente de gradient a pour but de trouver les paramètres optimaux (a1...an) tel que la somme des erreurs entre la sortie attendue yi et la sortie obtenue F(Xi) soit la plus petite possible.

Clic3
Comment ça fonctionne ? On part d'un point initial qui représente une certaine valeur de nos paramètres (a1...an) et on calcule par récurrence, avec la formule ici, la valeur des paramètres à l'itération suivante jusqu'à convergence de l'algorithme.

Dans une descente de gradient classique, la fonction f est l'erreur totale E. Donc à chaque itération on utilise toutes les données, ce qui peut être très long. Dans une descente de gradient stochastique, la fonction f est une erreur Ei sélectionnée au hasard, et donc à chaque itération, on utilise une seule donnée i. Lorsque toutes les données ont servi 1 fois (lorsque nous avons réalisé un epoch), on réalise un autre epoch, et ce jusqu'à convergence de l'algorithme.

On va utiliser méthode stochastique.

Diapo 17
Clic1
Dans notre cas, les erreurs à minimiser sont les erreurs élémentaires eui

Clic2
On ajoute à cette expression des termes de régularisation qui améliorent les performances de l'algorithme en évitant le surapprentissage.

On ajoute également des termes de biais :
- bu : le biais des utilisateurs
- bi : le biais des items
- et la moyenne globale des notes attribuées mu.

Clic3
Les paramètres mis à jour à chaque itération sont les coefficients de P et Q et les termes de biais.

Clic4
Avec la formule du gradient précédente, nous pouvons alors calculer la relation de récurrence pour ces paramètres.

Diapo 18
Clic1
L'algorithme implémenté commence par initialiser les matrices P et Q avec des valeurs aléatoires, les biais à zéro et la moyenne globale avec l'expression ci-contre.

Clic2
Nous créons ensuite une liste de triplets contenant les utilisateurs u, items i, et notes réelles de u pour i.

Clic3
Pour nb_epochs itérations :
- on mélange le dataset ce qui revient à choisir une erreur Ei au hasard.

Clic4
- on réalise 1 epoch : on met à jour nos paramètres en utilisant au moins une fois toutes nos données

****
****

Diapo 19
Clic1
Concernant les paramètres de notre fonction, on fixe beta à 0.02,

Clic2
et nb_epochs à 10

Clic3
Le nombre de caractéristiques latentes K est déterminé empiriquement et est égal au nombre de valeurs singulières dominantes de la matrice des notations R. On trouve une valeur de 200 pour le premier jeu de données puis autour de 15 pour les autres jeux de données.

Clic4
Le choix du taux de convergence α estimé empiriquement est important car il détermine si l'algorithme converge ou non. On trouve des valeurs entre 10^{-2} et 10^{-4}**.**

Diapo 20
Clic1
Après calcul de la MAE et de la RMSE pour les 4 jeux de données, nous obtenons une MAE autour de 0.7 pour le premier jeu et entre 3.5 et 4 pour les autres et une RMSE autour de 0.9 ou 4.9

Clic2
Etonnamment, l'algorithme de descente de gradient est moins performant que le prédicteur de base ou l'algorithme des plus proches voisins.

Clic3
En effet, les algorithmes de factorisation matricielle donnent de meilleurs résultats lorsque le jeu de données devient conséquent. A ce stade, l'espace de stockage devient insuffisant pour les algorithmes basés sur la mémoire, et il y a suffisamment de données pour que les caractéristiques latentes deviennent très apparentes.

Clic4
Si on supprime des jeux de données les utilisateurs qui n'ont pas noté d'item ou les items qui n'ont pas donné de notes

Clic5
nous obtenons de meilleurs résultats pour les 3 premiers jeux de données.

Clic6
Un des défauts de l'algorithme de Funk-SVD est donc qu'il donne de moins bons résultats lorsqu'il y a des notations manquantes dans le jeu de données, ce qui est souvent le cas en réalité.

Clic7
Ce défaut peut être compensé par la grande quantité de donnée puisqu'on constate que seul le dernier jeu de données donne de moins bons résultats lorsqu'on supprime les utilisateurs ou items sans note.

5 min

5 - ALS :

Diapo 21- Une 2e méthode de factorisation matricielle consiste à utiliser l'algorithme des moindres carrés alternés dit ALS.

Il existe 3 principaux avantages à réaliser un algorithme de moindres carrés alternés. Tout d'abord, il s'avère que contrairement aux algorithmes tels que le KNN ou encore le baseline predictor, l'ALS est, comme la Funk SVD, adapté aux grands jeux de données. Ensuite, cet algorithme est aussi adapté dans le cas où le jeu de données contient très peu de notes par

rapport à la quantité d'utilisateurs et d'items. Enfin, l'ALS est un algorithme qui s'implémente plus facilement que ceux que nous avons vu précédemment et propose aussi des résultats en peu de temps.

Diapo 22- L'objectif ici est similaire à celui vu pour la Funk SVD. En effet, nous voulons ici approximer notre matrice R

Clic en un produit de 2 matrices de rangs réduits U^T par M.

Clic Avec U et M les matrices représentées ici. U et M sont des matrices respectivement de taille Kxm et Kxn. U représente en quelque sorte les utilisateurs alors que M représente les items. Quant au nombre de facteurs latents K, il joue le même rôle que pour la Funk SVD à savoir qu'il tente d'expliquer les interactions observées entre un grand nombre d'utilisateurs et d'items par un nombre relativement faible de facteurs sous-jacents non observés.

****
****

Diapo 23– Finalement en formulant notre problème, on voit que si l'on veut R quasi égal à R tilde, cela revient avec nos notations, à minimiser l'écart entre la note de i donnée par l'utilisateur u et le produit scalaire des vecteurs u_u par m_i. En intégrant le terme de régularisation cela revient donc à minimiser la fonction perte écrite ici.

Clic Le premier terme correspond donc à la somme des erreurs au carré que nous venons de citer et

Clic le second terme correspond au terme de régularisation qui consiste à éviter le surajustement.

Clic Nous nous ramenons donc à un problème d'optimisation. Après avoir prouvé l'existence et l'unicité du minimum de cette fonction perte, nous pouvons nous lancer dans la recherche des points critiques qui sont alors les seuls candidats pour être le minimum de notre fonction.

Il s'agira en fait comme l'indique le nom de notre algorithme, de résoudre alternativement un problème de moindres carrés, d'une part pour déterminer chaque vecteur u_u en considérant les vecteurs m_i constants et de la même manière déterminer chaque vecteur m_i en considérant les vecteurs u_u constants.

Diapo 24 - Après calculs, on trouve le résultat suivant : D'une part le calcul de chaque colonne de la matrice U est donnée par la formule : avec (lire les matrice A_u et V_u). La matrice MIu contient en fait les colonnes représentant les items notés par l'utilisateur u. I_u stocke alors les identifiants des items alors notés par u. Enfin on note R(u, Iu) le vecteur ligne contenant chaque note donnée à l'item it_i par l'utilisateur u. En considérant U constante,

Clic on remarque que les colonnes de la matrice M sont calculées avec un raisonnement similaire.

###FACULTATIF

Diapo 25– Concernant maintenant l'implémentation de notre algorithme, nous avons écrit 2 fonctions : la première Init_M permettant d'initialiser notre matrice M. Puis la fonction principale Alternating_Least_Square qui prend 4 arguments : la matrice d'entrainement R_train, le paramètre de régularisation, K, et enfin le nombre maximal d'itérations it_max qui permet à notre algorithme de ne pas continuer à tourner alors que les calculs restants ne sont pas utiles.

Notre algorithme se décompose en 3 étapes : Initialisation des matrices U et M puis les mises à jour de U et M alternativement jusqu'à ce que nos critères d'arrêts soient vérifiés à savoir : le nb d'itérations inférieur au nb max d'itérations ou encore l'écart entre les RMSE de l'étape courante avec la précédente qui est inférieur en valeur absolue à 10^-4. ###

****
****

Diapo 26– Enfin nous pouvons nous intéresser à la performance de notre algorithme sur les jeux de données qui sont à notre disposition. Tout d'abord, commençons par movielens. Les paramètres optimisant le RMSE ont été trouvés de manière empirique. Pour le paramètre de régularisation lambda on a tracé sur le graphe de gauche la RMSE en fonction de lambda pour plusieurs valeurs de K. Globalement on trouve un lambda optimal égal à 0.16. Concernant K, on trouve la valeur 100, car à partir de cette valeur, la RMSE reste faible et stable. Pour ces valeurs de lambda et K, la RMSE et MAE sont similaires aux pgrm précédents cad 0.88 et 0.68 ce qui est plutôt satisfaisant.

Diapo 27– Ensuite on s'intéresse aux différents jeux de données Jester. Pour les paramètres optimaux indiqués sur ce tableau on observe un RMSE d'environ égal à 4 ce qui est meilleur que les programmes vus précédemment. Cela est cohérent étant donné que ce pgrm de moindres carrés alternés est plus adapté aux grandes bases de données.

****
****

6 - Conclusion

Diapo 28