Factorisation matricielle pour la recommendation[1]

Xuedong Shang

26 mai 2015

Table des matières

Résumé

La société Netflix a organisé en 2007 une compétition de recommandation mettant en jeu un million de dollars, afin de trouver le meilleur algorithme permettant de prédire au mieux le vote des utilisateurs. Cet article nous a proposé donc une réalisation assez puissante autour de cet événement. Il s'agit d'une méthode de factorisation qui cherche à extraire des variables latentes à la fois pour les utilisateurs et pour les produits.

Mots-clés. Apprentissage, factorisation matricielle, filtrage collaboratif, descente de gradient stochastique, moindres carrés.

1 Introduction

Le problème de recommandation consiste, pour un utilisateur donné, à prédire son score pour une certaine liste d'articles, afin que le revendeur puisse lui donner une recommandation personnalisée.

Il existe principalement deux grandes classes d'approches pour le problème de recommandation. Les premières sont les approches orientées contenu qui consistent à recommander des articles proches à ceux précédemment appréciés par les utilisateurs donnés. L'inconvénient important de cette approche est que des informations externes sont nécessaires mais sont parfois pas facile à trouver.

Une approche alternative est celle de type filtrage collaboratif. Il s'agit de recommander à un utilisateur donné des articles que d'autres utilisateurs ont appréciés. Dans notre article, nous allons nous concentrer sur l'une des méthodes classiques qui fait partie de type collaboritif, dite factorisation matricielle. Nous allons d'abord décrire le modèle de base et des algorithmes d'apprentissage utilisés. Ensuite, on va essayer d'améliorer la performance de notre modèle en y rajoutant des informations supplémentaires.

2 Le modèle

L'idée principale de la factorisation matricielle est très intuitive, c'est-à-dire de trouver deux matrices dont la multiplication est égale à la matrice originelle. Cela nous permet de trouver des variables latentes cachées entre deux différents objets, dans notre cas, ça sera les utilisateurs et les articles. Ces variables latentes jouent des différents rôles lorsque un utilisateur choisit son article. Par exemple, deux utilisateurs qui sont fans du même acteur auront probablement la tendance à donner un score remarquable au même film.

2.1 Un peu de maths

Passons maintenant à des choses plus théoriques.

Supposons que l'on dispose d'une matrice de scores R, un ensemble d'utilisateurs U et un ensemble d'articles I. Supposons aussi que l'on dispose d'un ensemble de variables latentes F. Notre objectif est donc de trouver deux matrice P, correspondant aux utilisateurs, et Q, correspondant aux articles, P sera de taille $|U| \times |F|$, et Q sera de taille $|I| \times |F|$ telles que :

$$\hat{R} = P \times Q^T \approx R$$

Ainsi la prédiction du vote de l'article q_i donné par l'utilisateur p_u est tout simplement le produit scalaire des vectors associés à p_u et q_i :

$$\hat{r}_{ui} = p_u^T q_i$$

Dans des travaux précedents, on utilise souvent l'imputation pour remplir la matrice creusée, mais cela peut causer des surapprentissages. Donc, dans notre modèle, on ne va utiliser que des données effectives, en y rajoutant un terme de régularisation pour éviter la surapprentissage. Ainsi on cherche à minimiser l'erreur quadratique moyenne :

$$\sum_{(u,i)\in\kappa} (r_{ui} - \sum_{f=1}^{F} p_{uf} q_{fi})^2 + \lambda(||p_u||^2 + ||q_i||^2)$$

Ici, κ désigne l'ensemble des couples (u, i) où r_{ui} est connu. Le terme de régularisation est souvent determinée par la validation croisée.

2.2 Algorithmes d'apprentissage

Il y a deux méthodes pour minimiser l'équation précédente.

2.2.1 Descente de gradient stochastique

La descente de gradient stochastique est l'approche plus populaire pour minimiser notre fonctionnelle.

On pose $e_{ui}^2 = (r_{ui} - \sum_{f=1}^{F} p_{uf} q_{fi})^2 + \lambda \sum_{f=1}^{F} (||p_{uf}||^2 + ||q_{fi}||^2)$. Pour minimiser l'erreur, il faut décider dans quelle direction on va modifier la valeur de p_{uf} et q_{fi}, ainsi on doit calculer les dérivées partielles de l'expression précédente par rapport à chaque variable p_{uf} et q_{fi}. On a :

$$\frac{\partial}{\partial p_{uf}} e_{ui}^2 = -2(r_{ui} - \hat{r}_{ui})(q_{fi}) + 2\lambda p_{uf} = -2e_{ui} q_{fi} + 2\lambda p_{uf}$$

$$\frac{\partial}{\partial q_{fi}} e_{ui}^2 = -2(r_{ui} - \hat{r}_{ui})(p_{uf}) + 2\lambda q_{fi} = -2e_{ui} p_{uf} + 2\lambda q_{fi}$$

Par conséquent, on met à jour les variables p_{uf} et q_{fi} dans la direction opposée à celle du gradient avec un pas γ :

$$p_{uf} \leftarrow p_{uf} + \gamma(e_{ui} q_{fi} - \lambda p_{uf})$$

$$q_{fi} \leftarrow q_{fi} + \gamma(e_{ui} p_{uf} - \lambda q_{fi})$$

2.2.2 Moindres carrés

Un algorithme alternatif est d'appliquer l'algorithme des moindres carrés par rapport aux p_u en fixant les q_i, puis échanger leur rôles et appliquer de nouveau cet algorithme.

Cette approche est particulièrement efficace lorsque la parallélisation est appliquée ou quand il s'agit d'une matrice de votes non creusée.

2.3 Améliorations

La factorisation matricielle est une approche assez flexible, c'est-à-dire que l'on peut y rajouter des différents aspects de données afin qu'elle puisse s'adapter à des différentes applications.

2.3.1 Ajout des biais

Le premier élément que l'on va étudier sont des biais. Les biais sont des facteurs qui ne concernent soit les utilisateurs, soit les produits.

Prenons un exemple pour illustrer cette notion. Disons que la note moyenne de tous les films sur Netflix est 3/5. Le film Titanic est un film qui est considéré comme un bon film, donc recevra en général une note plus haute que la moyenne, disons que 3,5/5, ainsi le 0,5 est le biais pour cet article. D'autre côté, moi je suis un spectateur strict, je donne souvent 1 point de moins que la moyenne, ainsi le 1 est le biais pour moi en tant que utilisateur. Par conséquent, la note que je donne à Titanic sera $3+0,5-1=2,5$.

On peut formaliser ce propos facilement. Posons μ la moyenne globale, b_i le biais pour l'article i, b_u pour l'utilisateur u, alors la nouvelle note approchée sera donnée par la formule suivante :

$$\hat{r}_{ui} = \mu + b_i + b_u + p_u^T q_i$$

La nouvelle fonctionnelle à minimiser devient donc :

$$\sum_{(u,i)\in\kappa} (r_{ui} - \mu - b_i - b_u - \sum_{f=1}^{F} p_{uf} q_{fi})^2 + \lambda(||p_u||^2 + ||q_i||^2 + b_i^2 + b_u^2)$$

2.3.2 Des entrées supplémentaires

À venir.

2.3.3 Dynamiques temporelles

À venir.

2.3.4 Des entrées avec des niveaux de confidence variés

À venir.

3 Les résultats

À venir.

4 Conclusion

Références

[1] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *IEEE Computer Society*, (0018-9162) :42–49, 2009.