Introduction to Recommendation Systems with Collaborative Filtering

AI Frameworks

Brendan Guillouet

6th December 2020

Institut National des Sciences Appliquées

Introduction

Table of contents

E-commerce and recommandation system

Context

- Customer relationship management.
- Appetence score.
  - Google double click, Criteo.
- Product recommendation

Recommendation system

- Content Base. Using and Item metadata.
- Adaptive filtering (Bandit).
- Collaborative filtering.
  - Based on existing product X client relationship

2 types of solution.

- Based on neighborhood.
- Based on latent factor.
  - Matrices factorization.
  - Matrices completion
  - Neural Network

Difficulties

- How to evaluate the recommendation ?
- Number of parameters to estimate.
  - Cold start, number of latent factors etc..

Neighborhood-based methods

Two methods

1. User-User Filter
   - Find a similarity metric between users.
   - For a user u, find the set of k closest users S_u^k.
   - Predict a rate, $\widehat{r_{u,i}}$ for an item i as a linear combination of known rates $r_{u',i}$ for $u' \in S_u^k$.

Two methods

1. User-User Filter
    - Find a similarity metric between users.
    - For a user u, find the set of k closest users S_u^k.
    - Predict a rate, $\widehat{r_{u,i}}$ for an item i as a linear combination of known rates $r_{u',i}$ for $u' \in S_u^k$.
2. Item-Item Filter
    - Find a similarity between items.
    - For a user i, find the set of k closest users S_i^k.
    - Predict a rate, $\widehat{r_{u,i}}$ for an user u as a linear combination of known rates $r_{u,i'}$ for $i' \in S_i^k$

Main assumption : customers with a similar profile will have similar tastes.

Generic formula.

$$\widehat{r}_{u,i} = \frac{\sum_{u' \in S_u^k} s(u,u') \cdot r_{u',i}}{\sum_{u' \in S_u^k} |s(u,u')|}$$

- $r_{u,i}$ is the rate given by user u to item i.
- s is a proximity score between u and u'.
- S_u^k is the set of k closest neighbors of user u

Main assumption : customers with a similar profile will have similar tastes.

Basic formula with mean.

$$\widehat{r}_{u,j} = \bar{r}_{u,.} + \frac{\sum_{u' \in S_u^k} s(u,u') \cdot (r_{u',i} - \bar{r_{u',.}})}{\sum_{u' \in S_u^k} |s(u,u')|}$$

- $r_{u,i}$ is the rate given by user u to item i.
- s is a proximity score between u and u'.
- S_u^k is the set of k closest neighbors of user u
- $\bar{r}_{u,.}$ is the mean rate given by user u

Main assumption : customers with a similar profile will have similar tastes.

Basic formula with Z-score.

$$\widehat{r}_{u,j} = \bar{r}_{u,.} + \sigma_{u,.} \cdot \frac{\sum_{u' \in S_u^k} s(u,u') \cdot \frac{(r_{u',i} - \bar{r_{u',.}})}{\sigma_{u',.}}}{\sum_{u' \in S_u^k} |s(u,u')|}$$

- $r_{u,i}$ is the rate given by user u to item i.
- s is a proximity score between u and u'.
- S_u^k is the set of k closest neighbors of user u
- $\bar{r}_{u,.}$ is the mean rate given by user u
- $\sigma_{u,.}$ is the standard deviation of rate given by user u

Pearson correlation coefficient

$$s(u,v) = \frac{\sum_{i \in I_u \cap I_v} (r_{u,i} - \bar{r_u}) \cdot (r_{v,i} - \bar{r_v})}{\sqrt{\sum_{i \in I_u \cap I_v} (r_{u,i} - \bar{r_u})^2} \cdot \sqrt{\sum_{i \in I_u \cap I_v} (r_{v,i} - \bar{r_v})^2}}$$

Spearman rank correlation coefficient

$$s(u,v) = \frac{\sum_{i \in I_u \cap I_v} (\tilde{r_{u,i}} - \bar{\tilde{r_u}}) \cdot (\tilde{r_{v,i}} - \bar{\tilde{r_v}})}{\sqrt{\sum_{i \in I_u \cap I_v} (\tilde{r_{u,i}} - \bar{\tilde{r_u}})^2} \cdot \sqrt{\sum_{i \in I_u \cap I_v} (\tilde{r_{v,i}} - \bar{\tilde{r_v}})^2}}$$

$\tilde{r_{u,i}}$ is the rank of product i in the preferences of user u (the higher the score, the smaller the rank).

Cosine : based on vectorial space structure

$$s(u,v) = cos(u,v) = \frac{< u, v >}{||u|| \cdot ||v||}$$

The unknown values have to be filled with zeros to compute this score.

Main assumption : the customers will prefer products that share a high similarity with those already well appreciated.

Generic formula.

$$\widehat{r}_{u,i} = \frac{\sum_{i' \in S_i^k} s(i,i') \cdot r_{u,i'}}{\sum_{i' \in S_i^k} |s(i,i')|}$$

- $r_{u,i}$ is the rate given by user u to item i.
- s is a proximity score between i and i'.
- S_i^k is the set of k closest neighbors of item i.

Main assumption : the customers will prefer products that share a high similarity with those already well appreciated.

BASIC FORMULA WITH MEAN.

$$\widehat{r}_{u,i} = \bar{r}_{.,i} + \frac{\sum_{i' \in S_i^k} s(i,i') \cdot (r_{u,i'} - \bar{r}_{.,i'})}{\sum_{i' \in S_i^k} |s(i,i')|}$$

- $r_{u,i}$ is the rate given by user u to item i.
- s is a proximity score between i and i'.
- S_i^k is the set of k closest neighbors of item i.
- $\bar{r}_{.,i}$ is the mean rate given to item i

Main assumption : the customers will prefer products that share a high similarity with those already well appreciated.

BASIC FORMULA WITH Z-SCORE.

$$\hat{r}_{u,i} = \bar{r}_{.,i} + \sigma_{.,i} \cdot \frac{\sum_{i' \in S_i^k} s(i,i') \cdot \frac{(r_{u,i'} - \bar{r}_{.,i'})}{\sigma_{.,i'}}}{\sum_{i' \in S_i^k} |s(i,i')|}$$

- $r_{u,i}$ is the rate given by user u to item i.
- s is a proximity score between i and i'.
- S_i^k is the set of k closest neighbors of item i.
- $\bar{r}_{.,i}$ is the mean rate given to item i
- $\sigma_{.,i}$ is the standard deviation of rate given to item i

Same proximity score can be used :

Pearson correlation coefficient

Spearman rank correlation coefficient

Cosine : based on vectorial space structure

Another approach relies on the Bayesian paradigm.

Conditional probability

$$s(i,j) = P[j \in B | iB]$$

Where B is the purchase history.

- Easy to interpret results.
- Choice of k can be done with cross-validation.
- Good performance on small dataset..
- … but suffer from scalability problem as user base grows.
- Use with Surprise python library.

Latent Vector-based methods

Latent Vector-based methods

Factorization

Let $X \in \mathcal{M}_{N \times M}$ be the matrice of user/item notations where N is the number of users and M the number of items.

MAIN IDEA : Approximate X as a product of two matrices $W \in \mathcal{M}_{N \times K}$ and $H \in \mathcal{M}_{K \times M}$

items

users

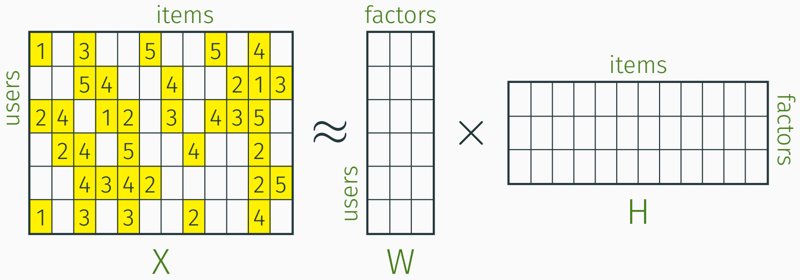
1		3			5			5		4	
		5	4			4			2	1	3
2	4		1	2		3		4	3	5	
	2	4		5			4			2	
		4	3	4	2					2	5
1		3		3			2			4	

X

Matrix Factorization

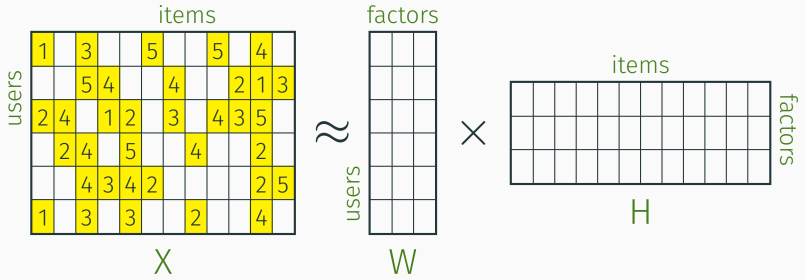
Let $X \in \mathcal{M}_{N \times M}$ be the matrice of user/item notations where N is the number of users and M the number of items.

Main Idea : Approximate X as a product of two matrices $W \in \mathcal{M}_{N \times K}$ and $H \in \mathcal{M}_{K \times M}$



Let $X \in \mathcal{M}_{N \times M}$ be the matrice of user/item notations where N is the number of users and M the number of items.

MAIN IDEA : Approximate X as a product of two matrices $W \in \mathcal{M}_{N \times K}$ and $H \in \mathcal{M}_{K \times M}$



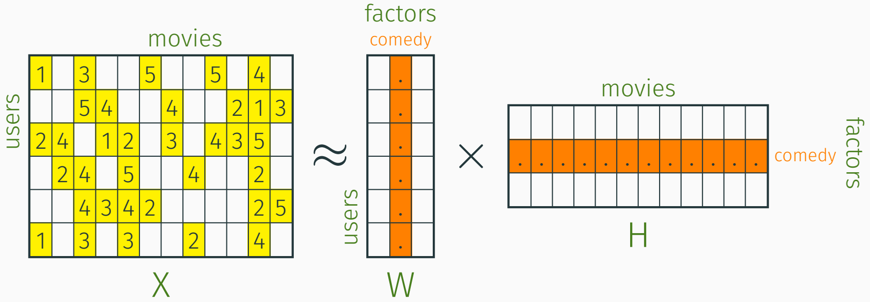
Each entry of X can be approximated with the following formula :

$$X_{i,j} = W_{i,.}^T H_{.,j} = \sum_{k \in K} w_{i,k} \cdot h_{k,j}$$

Interpretation

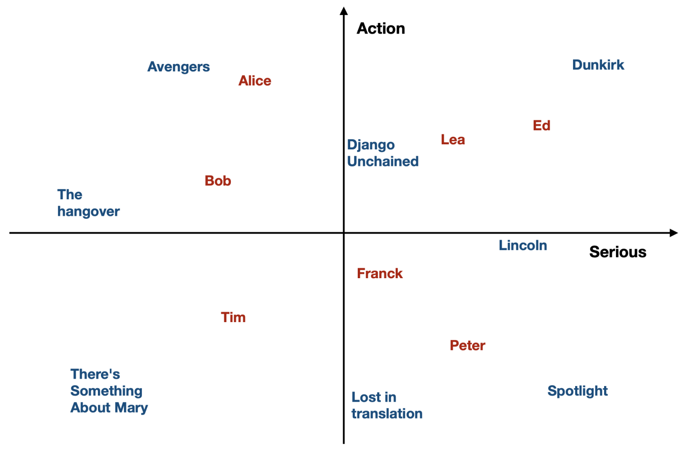
The K latent vectors can be interpreted as embedding.

Example : Movie recommendation



Latent factors can be interpreted a the genre of the movie.

Interpretation



Similar users/movies get embedded nearby in the embedding space . It's possible to compute similarities in this space.

To recover W and H the following minimization problem is introduced :

$$\min_{W,H\geq 0, rk(W)=r, rk(H)=r} L(X, WH) + \lambda_w P(W) + \lambda_h P(H)$$

- L is the loss function (measure accuracy of prediction)
- λ_w, λ_h are penalization parameter
- P is the penalty function.

To recover W and H the following minimization problem is introduced :

$$\min_{W,H \geq 0, rk(W)=r, rk(H)=r} L(X, WH) + \lambda_w P(W) + \lambda_h P(H)$$

- L is the loss function (measure accuracy of prediction)
- λ_w, λ_h are penalization parameter
- P is the penalty function.

Constraint of positivity on W and H $\Longrightarrow$ Non Negative Matrix Factorization.

NMF Algorithms

NMF algorithms generally solve problem iteratively.

Lee and Seung [1999] Euclidean loss, no regularization.

$$H_{k,j} \leftarrow H_{k,j} \frac{(W^T X)_{k,j}}{(W^T W H)_{k,j}}, \quad W_{i,k} \leftarrow W_{i,k} \frac{(X H^T)_{i,k}}{(V H H^T)_{i,k}},$$

Brunet et al. [2004] Kullback-Leibler loss, no regulatization.

$$H_{k,j} \leftarrow H_{k,j} \frac{\sum_l \frac{W_{l,k} X_{l,j}}{(WH)_{l,j}}}{\sum_l W_{l,k}}, \quad W_{i,k} \leftarrow W_{i,k} \frac{\sum_l \frac{H_{k,l} X_{i,l}}{(WH)_{i,l}}}{\sum_l H_{k,l}}$$

Most algorithm available in R NMF package are variation of those algorithms.

Example with L_2 regularization and euclidean loss function.

$$\min_{W,H\geq 0, rk(W)=r, rk(H)=r} \sum_{i,j}(x_{i,j} - w_i^T h_j)^2 + \sum_{i=1}^{N} \lambda_w ||w_i||^2 + \sum_{j=1}^{M} \lambda_h ||h_j||^2$$

A non convex problem.

Example with L_2 regularization and euclidean loss function.

$$\min_{W,H\geq 0, rk(W)=r, rk(H)=r} \sum_{i,j}(x_{i,j} - w_i^T h_j)^2 + \sum_{i=1}^{N} \lambda_w ||w_i||^2 + \sum_{j=1}^{M} \lambda_h ||h_j||^2$$

A non convex problem.

SOLUTION : Alternate Lleast Square.

$\forall i$, fix all variables except w_i and solve :

$$argmin_{w_i} \sum_{j}(x_{i,j} - w_i^T h_j)^2 + \lambda_w ||w_i||^2$$

Do the same with fix h_j, $\forall j$.

It is easy to show that :

$$argmin_{w_i} \sum_j (x_{i,j} - w_i^T h_j)^2 + \lambda_w ||w_i||^2$$

Has the solution

$$w_i = (\sum_j h_j h_j^T + \lambda_w I_k)^{-1} (\sum_j x_{i,j} v_j)$$

This is very similar to regularized least squares regression.

These algorithms has been widely studied and a lot of different implementation exist.

- **NMF** R package (9 implementation among Kim and Park [2007], Lee and Seung [1999], Brunet et al. [2004]).
  - get a lot of interpretation tools such (consensus matrix, dendogram etc..)
- **MLlib** Spark (Koren et al. [2009]).
  - Implementation that handle missing value.
- **Surprise** Python (Zhang et al. [2006], Luo et al. [2014]).
- **Scikit-learn** Python (Lee and Seung [1999], Hoyer [2004])

Remarks

- SVD decomposition can also provide a solutions for recomendation.

$$X = UDV^*$$

  - Good property as unity and optimal solution are guaranteed for fixed rank.
- Optimize the rank.
- NMF minimization problem does not always lead to well posed problem.
- Cold start problem not discussed but it's a major research point.
- Most of the implementation of these algorithm treat missing rates as zero.

Latent Vector-based methods

Completion

Mazumder et al. [2010] re-write the minimization problem as

$$\min_M \frac{1}{2}\|P_\Omega(X) - P_\Omega(M)\|_F^2 + \lambda\|M\|_*$$

where

- Ω contain the pairs of indices (i, j) where X is observed
- $P_\Omega(X)_{i,j} = X_{i,j}$ if $X_{i,j}$ is known, 0 otherwise.
- $\|M\|_*$ is the nuclear norm of M

If $\widehat{M}$ solves problem, then it satisfies the following stationarity condition :

$$\widehat{M} = S_\lambda(Z)$$

where

$$Z = P_\Omega(X) + P_{\Omega^\perp}(\widehat{M})$$

and

$$S_\lambda(Z) = UD_\lambda V', \quad D_\lambda = diag[(d_1 - \lambda)_+, \ldots, (d_r - \lambda)_+]$$

Reconstruct $S_\lambda(Z) = UD_\lambda V^T$ from SVD decomposition is called "soft-thresholded SVD".

For sufficiently large λ, D_λ will be rank-reduced, and hence so will be $UD_\lambda V^T$.

Pseudo Code

Initialize $\widehat{M}$

Iterate over the following steps :

- Compute $Z = P_{\Omega}(X) + P_{\Omega^{\perp}}(\widehat{M})$.
- Compute $\widehat{M} = S_{\lambda}(Z)$.

- We only use know value of X.
- Much more faster than NMF.
- implementation within the **softImpute** R package.

Latent Vector-based methods

Neural Network-based methods

As for NMF the main idea is minimze the objective function

$$\sum_{i,j \in \Omega} (r_{i,j} - w_i^T h_j)_2^2 + \lambda(||W||_2^2 + ||H||_2^2||)$$

where

- the embedding w_i of a movie i in an embedding space of size k.
- the embedding h_j of an item j in an embedding space of size k.

As for NMF the main idea is minimze the objective function

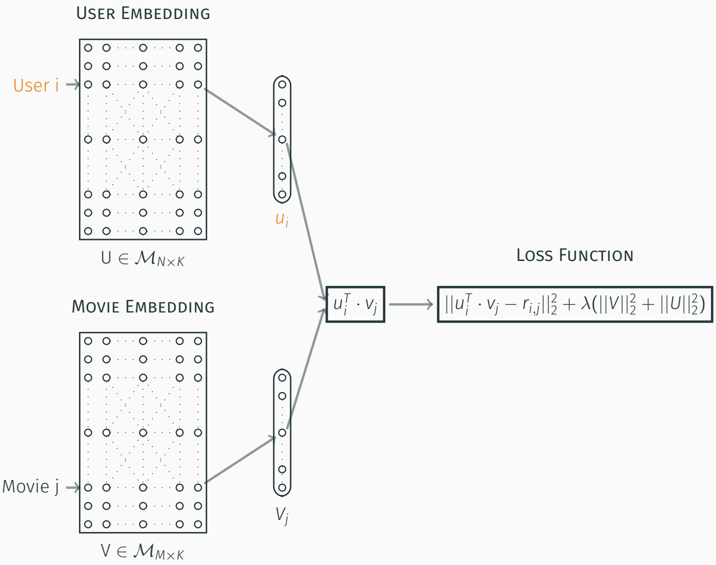
$$\sum_{i,j\in\Omega}(r_{i,j} - w_i^T h_j)_2^2 + \lambda(||W||_2^2 + ||H||_2^2||)$$

where

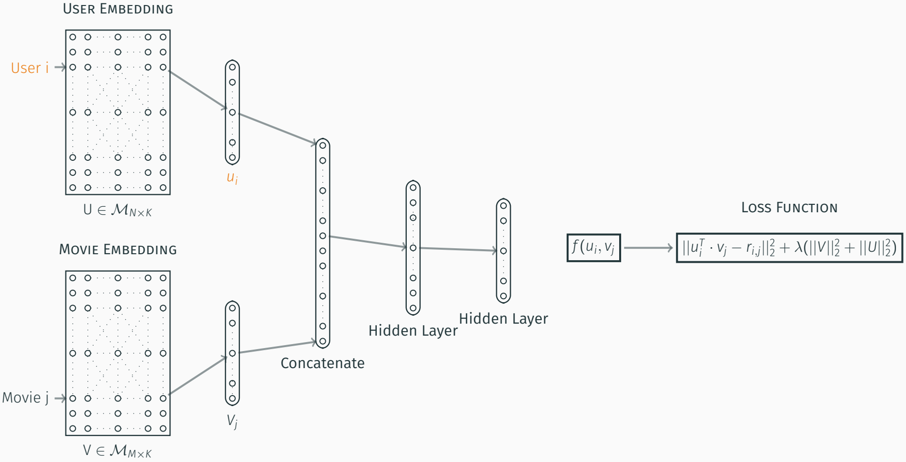
- the embedding w_i of a movie i in an embedding space of size k.
- the embedding h_j of an item j in an embedding space of size k.

Solution Use neural network architecture and framework to solve it!

Architecture



Deep Architecture



```
import tensorflow.keras.layer as kl
user_embedding = kl.Embedding(output_dim=embedding_size,
                              input_dim=input_dim,
                              input_length=1)
```

- embedding_size = K the size of latent vector
- input_dim = M The size of the user
- input_length = 1 (i,e the indice).

All weight of *user_embedding* are trainable and tuned during backprob.

Remarks

- Handle missing data (as it just optimize over existing user item couple
- Good performance comparing to other methods (especially if you have GPU).
- Hard to interpret the function you designed to compare the vector...
- ...but you can still interpret the embedding vectors!

TP

We will use the MovieLens dataset to compare the different methods.

- It's produced by the groupLens company (https://grouplens.org/).
- Various dataset :
  - Stable one. 25 Millions of rates. Use for challenge and benchmark.
  - Small one. 100K rates. Use for education or research. (Not stable).
- Contains various metadata that we won't use in this TP.
  - Genre, Tag, Age of user etc...

TP

1. Neighborhood-based methods
    - *1-Python-Neighborhood-MovieLens.ipynb*
        - Discover the data and surprise library. Apply Neighborhood based data.
2. Latent Vector-based methods
    - *2-R-Facto-MovieLens.ipynb*
        - NMF package on toy dataset. SoftImpute on movielens.
    - *2-Pyspark-Facto-MovieLens* (OPTIONAL)
    - *3-Python-Neural-MovieLens*
        - Apply Neural recommender system on movielens dataset. Visualize latent vector.

Reference

http://wikistat.fr/pdf/st-m-datSc3-colFil.pdf

https://cse.iitk.ac.in/users/piyush/courses/ml_autumn16/771A_lec14_slides.pdf

https://m2dsupsdlclass.github.io/lectures-labs/slides/03_recommender_systems/index.html

Références

Jean-Philippe Brunet, Pablo Tamayo, Todd R Golub, and Jill P Mesirov. Metagenes and molecular pattern discovery using matrix factorization. *Proceedings of the national academy of sciences*, 101(12) :4164–4169, 2004.

Patrik O Hoyer. Non-negative matrix factorization with sparseness constraints. *Journal of machine learning research*, 5(Nov) :1457–1469, 2004.

Hyunsoo Kim and Haesun Park. Sparse non-negative matrix factorizations via alternating non-negativity-constrained least squares for microarray data analysis. *Bioinformatics*, 23(12) :1495–1502, 2007.

Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, (8) :30–37, 2009.

Daniel D Lee and H Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755) :788, 1999.

Xin Luo, Mengchu Zhou, Yunni Xia, and Qingsheng Zhu. An efficient non-negative matrix-factorization-based approach to collaborative filtering for recommender systems. *IEEE Transactions on Industrial Informatics*, 10(2) :1273–1284, 2014.

Rahul Mazumder, Trevor Hastie, and Robert Tibshirani. Spectral regularization algorithms for learning large incomplete matrices. *Journal of machine learning research*, 11(Aug) :2287–2322, 2010.

Sheng Zhang, Weihong Wang, James Ford, and Fillia Makedon. Learning from incomplete ratings using non-negative matrix factorization. In *Proceedings of the 2006 SIAM international conference on data mining*, pages 549–553. SIAM, 2006.