Collaborative Filtering

Summary

We describe here the important principles of the collaborative filtering problem. This a common modern problem for www firms (amazon, netflix, movilens, ...). It involves unfortunately (or fortunately) non trivial mathematical derivations that will be briefly presented. This kind of problem also leads to complicated numerical tasks.

1 Introduction - Recommandation systems

1.1 Customer relationship management

The expansion rate of online e-commerce shops leads to very important human needs in marketing, especially for the CRM *customer relationship management*. It is maybe one of the most important massive data origin and the data-scientist must in that case produce a quantitative scoring. In general, the firms require a model that builds an appetance scoring for a product (see the example of the studies Premium credit card Visa Premier and life insurance products).

1.2 New challenges of on-line e-commerce

The on-line e-commerce introduces new challenges that will be referred to in what follows as collaborative filtering (or filtering for short).

The *filtering* problem studies one way to select and recommend automatically some goods or products to customers according to their former visits on the website of the firm, and to the informations store in the cookies of their browser. It is sometimes described as a recommandation system. [1]

In the next sessions, we will talk about two large families of problem dedicated to filtering.

- The first one concerns batched strategies : the data scientist possesses a large number of observations of the filtering problem and has to produce a recommendation for each new customer described by some partial observations (involved in the problem). Hence, the problem involves a traditionnal learning set and predictions should be considered as optimizers of the accuracy on a test set.
- The second one is more challenging and concerns on-line strategies : the data scientist must develop and algorithm that sequentially adapts its recommendation to a recursion of arrivals of new customers. It leads to the so-called Bandit's algorithms world.

1. The website site "PodcastScience" provides an introduction that is reasonnably complete on the problem.

	Batman Begins	*Alice in Wonderland*	*Dumb and Dumber*	*Equilibrium*
User A	4	?	3	5
User B	?	5	4	?
User C	5	4	2	?

FIGURE 1 – An example of movie recommandation problem.

In a sense, the major difference of these two problems can be illustrated as follows. Imagine $(X_1, \dots, X_n)$ is a set of n i.i.d. observations of the same law μ. We aim to recover $\mathbb{E}_{X\sim\mu}[X]$. The first approach relies on the batched estimate :

$$\hat{m}_n = \frac{1}{n}\sum_{j=1}^{n} X_j,$$

which consistently estimates $\mathbb{E}_{X\sim\mu}[X]$ under mild assumptions.

The second approach considers some sequential arrivals of the observations $(X_j)_{j\geq 1}$ and chooses to update from time to time its belief on $\mathbb{E}_{X\sim\mu}[X]$ with a recursive formula :

$$\hat{m}_n = \hat{m}_{n-1} + \gamma_n h(\hat{m}_{n-1}, X_n).$$

For example, in the case of the estimation of $\mathbb{E}_{X\sim\mu}[X]$, it is easy to see that

$$\hat{m}_n = \hat{m}_{n-1} + \frac{1}{n}[X_n - \hat{m}_{n-1}].$$

This formula looks like an explicit Euler step associated to a discretisation of an ordinary differential equation, which is sometimes a gradient descent. Here, it is the case while considering $H(m) = \mathbb{E}_{X\sim\mu}[X - m]^2$ (see the Lecture 4).

1.3 Collaborative filtering

This problem has beeen largely popularized by the Netflix challenge : it consists in finding a good proposal for a customer while observing its taste on movies seen before. The tastes of the customers are evaluated by a mark between 1 and 5 on each movie. Hence, the objective is to predict the mark of each customer on unseen movies, unread books, ... Most of the e-commerce important shops are running some algorithms of collaborative filtering (Amazon, Fnac, Netflix, Youtube...).

Warning : Collaborative filtering algorithms are only based on the interactions between customers and products (and not on additional informations).

The statistical methods generally belong to two large families :

- Neighborhoods methods : very elementary and based on similarity scoring on customers (linear correlation, Spearman ranking...) or on products (see [2] and Sections 2 and 3).
- Latent factor models : a prior on a sparse representation of the problem is assumed.

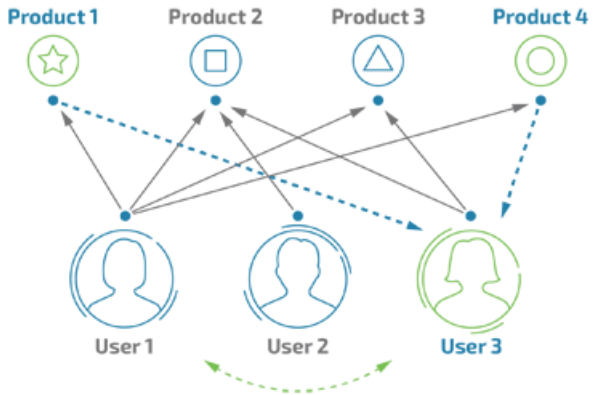

FIGURE 2 – Schematic representation of the user/user filtering approach.

2 User-User filters

Main assumption : customers with a similar profile will have similar tastes. For a customer u, the aim is to find a subset S_u of customers with a close profile and predicting the missing mark of a product i on customer u relies on a convex linear aggregation of marks of customers in S_u.

2.1 A generic way to compute the prediction

If one aims to predict the mark on product i by customer u, the prediction is defined as

$$\hat{r}_{u,i} = \bar{r}_u + \frac{\sum_{u'\in S_u} s(u,u')(r_{u',i} - \bar{r}_{u'})}{\sum_{u'\in S_u} |s(u,u')|}, \qquad (1)$$

where $s(u,u')$ is a proximity score between u and u' and $r_{u',i}$ is the rate given by customer u' on product i. Using canonical notations, $\bar{r}_u$ refers to the average rate (on all the available products) given by customer u.

Remark. — Subtracting the users mean rating $\bar{r}_u$ compensates for differences in users' use of the rating scale (some users will tend to give higher ratings than others).

To run (1), it is first necessary to define a similarity score s, and second to choose a threshold in this score to keep the highest values and then determine the subset S_u. The size of S_u can be arbritrary, or data driven according to a cross validation strategy. The important question remains the computation of the scores.

2.2 Scores

DÉFINITION 1. — *[Pearson correlation coefficient] The Pearson score is defined as*

$$s_{Pearson}(u,v) = \frac{\sum_{i\in I_u\cap I_v}(r_{u,i}-\bar{r}_u)(r_{v,i}-\bar{r}_v)}{\sqrt{\sum_{i\in I_u\cap I_v}(r_{u,i}-\bar{r}_u)^2}\sqrt{\sum_{i\in I_u\cap I_v}(r_{v,i}-\bar{r}_v)^2}},$$

where $r_{u,i}$ refers to the rate obtained by product i for customer u. Of course, $\bar{r}_u$ is the average rate given by customer u.

DÉFINITION 2. — *[Spearman rank correlation coefficient] The Spearman rank correlation is computed as*

$$s_{Spearman}(u,v) = \frac{\sum_{i\in I_u\cap I_v}(\tilde{r}_{u,i}-\bar{\tilde{r}}_u)(\tilde{r}_{v,i}-\bar{\tilde{r}}_v)}{\sqrt{\sum_{i\in I_u\cap I_v}(\tilde{r}_{u,i}-\bar{\tilde{r}}_u)^2}\sqrt{\sum_{i\in I_u\cap I_v}(\tilde{r}_{v,i}-\bar{\tilde{r}}_v)^2}},$$

where $\tilde{r}_{u,i}$ refers to the rank of product i in the preferences of customer i (the higher the score, the smaller the rank).

DÉFINITION 3. — *[Cosine similarity] This coefficient is based on a vectorial space structure, it is computed as*

$$s_{cosine}(u,v) := \cos(u,v) = \frac{\langle u,v\rangle}{\|u\|.\|v\|},$$

where the unknown values of u and v are filled by 0 to compute this score.

On the very simple example of Figure 1, we aim to predict the score of customer C on movie e. The average known rate of C is 3.667 and we use a Pearson correlation coefficient with a neighborhood size of 2. We can check that $s(C,A) = 0.832$ and $S(C,D) = -0.515$, leading to a predicted score of

$$\hat{r}_{C,e} = 4.667.$$

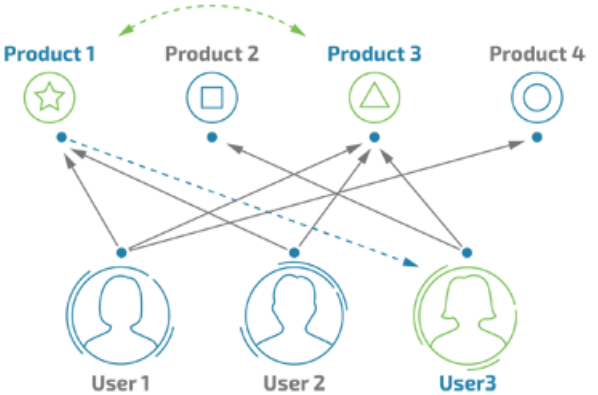

FIGURE 3 – Schematic representation of the Item/Item filtering approach.

2.3 Drawbacks

User-user collaborative filtering, while effective, suffers from scalability problems as the user base grows. Searching for the neighbors of a user is an O(|Ul) operation (or worse, depending on how similarities are computing ? directly computing most similarity functions against all other users is linear in the total number of ratings).

To extend collaborative filtering to large user bases and facilitate deployment on e-commerce shops, it was important to develop more scalable approaches. Rather than using similarities between users rating behavior to predict preferences, item- item filtering uses similarities between the rating patterns of items. If two items tend to have the same users like and dislike them, then they are similar and users are expected to have similar preferences for similar items. In its overall structure, therefore, this method is similar to earlier content-based approaches to recommendation and personalization, but item similarity is deduced from user preference patterns rather than extracted from item data.

3 Item Item filters

Main assumption : the customers will prefer products that share a high similarity with those already well appreciated. Prediction of product j : aggregate with a linear convex combination of products S_j that are closed to product j.

3.1 A generic way to compute the prediction

After collecting a set S of items similar to i, we can predict $r_{u,i}$ as follows :

$$\hat{r}_{u,i} := \frac{\sum_{j \in S} s(i,j) r_{u,j}}{\sum_{j \in S} |s(i,j)|}, \tag{2}$$

where $s(i,j)$ is a similarity score between product i and j whereas S is a set of products deduced from thresholding $s(i,.)$.

This last prediction suffers from certain problems of normalization. For further details, you are invited to read carefully the solutions proposed in [2].

3.1.1 Item similarity

The item-item prediction process requires an item ?item similarity matrix S.

DÉFINITION 4. — *[Cosine similarity] Cosine similarity between items is the most popular metric, as it is a simple fast and accurate estimator :*

$$s(i,j) := \frac{\langle r_i, r_j}{\|r_i\| \|r_j\|},$$

where again r_i is the vector of rates obtained by product i filled by 0 in missing values.

We can also propose to compute a Pearson correlation coefficient as it was done above for the user/user filtering. Another approach relies on the Bayesian paradigm.

DÉFINITION 5. — *[Conditional probability]*

$$s(i,j) := P[j \in \widehat{B|i} \in B]$$

where B refers to the user's purchase history.

On our simple example in Figure 1, we compute the rate of customer C on movie with a Cosine similarity and a neighborhood of size 2. We obtain

$$\begin{aligned} \hat{r}_{C,e} &= \frac{s(b,e) r_{C,b} + s(d,e) r_{C,d}}{|s(b,e)| + |s(d,e)|} \\ &= \frac{5 * 0.607 + 2 * 0.382}{0.607 + 0.382} \\ &= 3.84. \end{aligned}$$

4 Non negative Matrix Factorisation

The so-called NMF method for collaborative filtering relies on a matricial formulation of the problem. We call X the matrix of size $n \times p$ where n is the number of customer and p the number of items. Before saying something about its use for collaborative filtering, let us now provide briefly some insights on this factorization.

4.1 SVD

There exists a very common factorization of X which is called the SVD. This method involves a particular diagonal structure on one of the two matrices, and a orthogonality constraint on the rows of the second. The mathematical result states that if $X \in M_{n,p}(\mathbb{R})$ then

$$X = UDV^*,$$

where U is a unitary matrix of size $n \times n$, D is a $n \times p$ matrix with a non negative diagonal, and V^* is the adjoint matrix of unitary matrix of size $p \times p$.

4.2 NMF

The NMF is an alternative matrix factorization. X has to be approximated by $W \times H$ where we are looking for a sparse structure on the data. This sparsity is imposed by choosing $W \in M_{n,r}(\mathbb{R})$ and $H \in M_{r,p}(\mathbb{R})$, where $r << n \wedge p$. Moreover, we impose the constraint of **positivity** on W and H.

We can interpret the result of the NMF as follows : r is a list of latent factors that are not observed and should be recover. It can be thought as a number of

possible profiles of customers/clients couple.The real number $w_{i,j}$ is an appetance score for customer i to the latent factor j. Finally, $h_{j,k}$ represents the proportion of item k involved in the latent factor j.

To recover both W and H, we introduce the minimization problem

$$\min_{W,H\geq 0, rk(W)=r, rk(H)=r} L(X, WH) + \lambda_1 P(W) + \lambda_2 P(H), \tag{3}$$

where (λ_1, λ_2) are penalization parameters and $P(.)$ is a penalty function inducing sparsity. The function L measures the accuracy of the prediction of X by WH. It can involve the Frobenius norm or the Kullback-Leibler divergence :

$$L_{Frobenius}(A, B) = Tr((A-B)^t(A-B))$$

and

$$L_{KL}(A, B) = \sum_{i,j} [A_{i,j} \log(A_{i,j}/B_{i,j}) - X_{i,j} + B_{i,j}]$$

Note that the minimization of (3) does not always lead to a well posed problem since every matrix D of size $r \times r$ with non negative coefficients verifies

$$X \simeq WD^{-1}DH.$$

However, some efficient algorithms may be found (essentially projected gradient descent and alternate least square (ALS) minimization). The last approach is based on the remark that when H is fixed, the problem is convex on W and can be solved by semidefinite programming. Fortunately, the NMF package of R propose 11 methods, among them 9 are based on the initial contributions of Lee and Seung [3].

4.3 Application to recommandation

Assume now that X possesses some missing values, denoted " ?" in what follows. A natural extension to the variationnal minimization problem (3) is

$$\min_{W\geq 0, H\geq 0, rk(W)\wedge rk(H)\leq r} \sum_{X_{u,v}\neq ?} [X_{u,v} - L_u R_v]^2 \tag{4}$$

An alternative formulation can be though as follows :

$$\min_{Z: \sum_{X_{i,j}\neq ?} [X_{u,v} - Z_{u,v}]^2 \leq \delta} rk(Z). \tag{5}$$

4.4 A brief toy example

Follow the roadmap of `Section 2`

5 Convex relaxation of the rank function

5.1 Relaxation

We end the lecture with some considerations on the model involved in (3). If we think about the Lasso relaxation of the ℓ_0 norm, it may be possible to obtain a similar relaxation for the rank function, since it is the may source of "non convexity". Recall first the "dual" criterion described above

$$\min_{Z: \sum_{X_{i,j}\neq ?} [X_{u,v} - Z_{u,v}]^2 \leq \delta} rk(Z). \tag{6}$$

Even intractable, [1] shows that it is possible to recover efficiently the missing values of X with such a low rank factorization of X as soon as the dimensionality is balanced as follows : the number of observed values should not be too small :

$$|(i,j) \ : \ X_{i,j} \neq ?| >> kn^{1.2} \log(n).$$

But the problem is still NP hard. A recent work [4] proposes to use the convex enveloppe function of the rk function. This function is given by the so-called nuclear norm, denoted $\|Z\|_*$:

$$\|Z\|_* := \sum_{\lambda \in Sp(Z)} \lambda = Tr(\sqrt{Z^*Z}),$$

where $\sqrt{Z^*Z}$ is the positive semidefinite matrix B such that $B^2 = Z^*Z$. The important point is that this nuclear norm is *convex*, and permits to use the machinery of convex optimization toolbox for obtaining reasonnably good statistical algorithms in efficient computation time.

We refer to the Soft Impute method introduced in [4] for an implementation of the following convex program :

$$\min_{Z} \sum_{X_{u,v}\neq ?} [X_{u,v} - Z_{u,v}]^2 + \lambda \|Z\|_*. \tag{7}$$

Note that this looks very similar to the variationnal formulation of the Lasso. The penalty term is governed by λ, a positive coefficient that induces the sparsity of the model. The highest λ, the smaller the rank of Z.

5.2 Soft-imput package

5.2.1 First round

A very brief example, which is stupid at the present moment...

Loading the package.

```
require(softImpute)
```

Simulating a pure uniformly random database with missing values.

```
set.seed(1011)
x=matrix(rnorm(30),6,5)
x[sample(1:30,10,replace=FALSE)]=NA
x
```

Using Soft-Impute package.

```
fits=softImpute(x,trace=TRUE,type="svd")
fits
```

Since this is a small matrix, it has solved it using repeated SVDs. There is no penalization here ($\lambda = 0$), and by default the rank was taken to be 2. Since there is no penalization, if the rank was given to be $\min(m, n)$, then there is no restriction, and any values for the missing data would give the same minimum loss of 0. In other words, either penalization, or a rank restriction (or both) are needed for sensible imputation.

We can also use the ALS approach (non convex problem).

```
fita=softImpute(x,trace=TRUE)
fita2=softImpute(x,rank.max=3,lambda=1.9,trace=TRUE)

fits2$d
```

We can impute the missing values using complete(), which returns the full matrix :

```
complete(x,fits2)
```

5.2.2 Second round

We are now interested in a situation where the completion makes sense, *i.e.*, when the matrix X is described by

$$X = WH,$$

where W is a $n \times r$ non-neagtive matrix and H is a $r \times p$ matrix, where r stands for the rank of the matrix X, that quantifies the degree of variability in the set of observations

```
set.seed(1978)
r=4
n=20
p=25
prop=0.01

w=matrix(abs(rnorm(80)),n,r)
H=matrix(abs(rnorm(100)),r,p)
X=w%*%H
Z=X
X[sample(1:n*p,floor(n*p*prop),replace=FALSE)]=NA
```

Let s try several possible factorizations with different possible values of r.

```
#SVD
fits2=softImpute(X,trace=TRUE,type="svd")
#Completion
Zpred=complete(X,fits2)
(Z-Zpred)/abs(Z)

#SVD
fits4=softImpute(X,rank=4,type="svd")
#Completion
Zpred=complete(X,fits4)
(Z-Zpred)/abs(Z)
```

Impressive improvement ! :-] Further details may be found in the Hastie's webpage https://web.stanford.edu/~hastie/swData/softImpute/vignette.html !

Références

[1] E.J. Candes et T. Tao, *The Power of Convex Relaxation : Near-Optimal Matrix Completion*, Information Theory, IEEE Transactions on **56** (2010), n° 5, 2053–2080.

[2] M. Ekstrand, J. Riedl et J. KonstanLee, *Collaborative Filtering Recommender Systems*, Foundations and Trends in Human-Computer Interaction **4** (2010), n° 2.

[3] D. Lee et S. Seung, *Learning the parts of objects by non-negative matrix factorization*, Nature (1999).

[4] R. Mazumder, T. Hastie et R. Tibshirani, *Spectral Regularization Algorithms for Learning Large Incomplete Matrices*, Journal of Machine Learning Research **11** (2010), 2287–2322, http://www.stanford.edu/~hastie/Papers/mazumder10a.pdf.