BNMF: Bayesian probabilistic model of non-negative factorization for collaborative filtering

Priscila Valdiviezo-Díaz, Antonio Hernando, Jesús Bobadilla

2017-10-02

Computer Science and Electronics Deparment
Universidad Técnica Particula de Loja, Ecuador
Universidad Politécnica de Madrid, España
pmvaldiviezo@utpl.edu.ec, {antonio.hernando, jesus.bobadilla}@upm.es

Abstract

This document shows the implementation of the non-negative factorization method for collaborative filtering, developed by Hernando, Bobadilla, & Ortega (2016). Details about features of the R package can be obtained from the documentation enclosed in it.

1. Introduction

This package implements a MF method called Bayesian Non-negative Matrix Factorization (BNMF) recently published by (Hernando, Bobadilla, & Ortega, 2016). This model consider the existence of K latent factors explaining the ratings that users make. In this way, a K dimensional vector a_u is associate for each user **u** and a K dimensional vector b_i is associate for each item **i**. In this model $a_{u,k}$ and $b_{k,i}$ are bound to lie within the range [0, 1]. Latent factors represent groups of users who share the same tastes in the system. In this way, the parameter K indicates the number of such groups in the database. The value $a_{u,k}$ represents the probability that user **u** belongs to the group **k** of users. The value $b_{k,i}$ represents the probability that users in the group **k** like item **i**.

The prediction of ratings $p_{u,i} = a_u * b_i$

Thus, the input of BNMF is represented by a matrix of ratings **R** (user * items), the number of gropus (latent factors), alpha and beta parameters for the learning algorithm. BNMF automaticly learns from the input data the model parameters.

2. BNMF

BNMF is an algorithm which compute the prediction the tastes of users in recommender systems based on factorizing the rating matrix into two non negative matrices, one related to items and another one related to users. The input of the algorithm is a matrix of rating **R** of dimension N*M, In which each user provides information about how much he likes some items. The following notation related to the rating matrix is used in this algorithm:

```
N     Number of users in the ratings matrix,
M     Number of items in the ratings matrix,
r     Rating that user u has made over the item i.
```

Parameters

In addition, some parameters must be setting by the system.

```
k     This parameter indicates the number of groups of users that the algorithm is going to find out
alfa  This parameter is related to the possibility of obtaining overlapping groups of users sharing
      the same tastes.
eta   This parameter is related to the amount of evidence that the algorithm requires to deduce that
      a group of users likes an item
```

Based on this, the algorithm compute some matrices of the parameters of learning, these are lambda, positive epsilon and negative epsilon.This are learned using a variational inference technique for calculating these parameters.

The initial state of the model, is set as follows:

1. Initialize randomly gamma values entered as a matrix
2. Initialize randomly the matrix of positive and negative epsilons

The values of these free parameters will change iterating from the Eqs. (6), (9) presented in (Hernando, Bobadilla, & Ortega, 2016) until that they convergence, this is, repeat until changes are not significant.

Then, the prediction of rating is calculated, multiplying the user vector ($a_{u,k}$) and item vector ($b_{k,i}$).

Output:

Based on the input and parameters, the algorithm outputs are:

```
au,k   Matrix associated to user
bk,i   Matrix associated to items
pu,i   Probability that the user u likes the item i.
```

3. Example of the usage of BNMF

In this section we apply BNMF to predict the rating of the user to items, and to obtain important calculations about the tastes of users through of matrices au.k and bk.i, which allow to finding out groups of users sharing the same taste. In this example we use the complete dataset of Movielens ml-latest-small, we split it into training and test. The dataset is transformed in a user*items matrix using the recommenderlab package.

Data Loading

First, let us load the library and check the data:

```
library(BNMF)
#For processing of ratings matrix
library(recommenderlab)
#loadData() #to download the dataset from the official site
#loading dataset that this package includes
data(MovieLensLatest)
ratings.dat<-MovieLensLatest
pctSplit<-0.7 #for training
splitData(pctSplit,ratings.dat)
```

Processing the training dataset

For procesing the training dataset which is a dataframe that include the rating of user to items, we using the instrucción realRatingMatrix of the recommenderlab packake to represent of rating matrices R.

```
#convert to rating matrix
tmpRealRatingMatrix <- as(ratings.trn, "realRatingMatrix")
R <- as(tmpRealRatingMatrix, "matrix")
R[is.na(R)] <- 0
R[is.nan(R)] <- 0
dim(R)
```

```
## [1]  671 4767
```

The matrix R contains 671 rows and 4767 columns, rows reference to users and columns reference to items.

Predicting ratings and getting of groups

Here we compute the algorithm outputs two matrices and the probability that the user u likes the item i.

```
#System setting parameters
k<-6  #Number of groups (latent factors)
alpha<-0.8 #Control of group overlap
eta<-5 #Evidence that a group of users likes an item
iter<-5  #setting number of iterations
output<-BNMF(iter,R,k,alpha,eta)
```

Processing Results

The output of the BNMF is a list whose fields are: "au.k", matrix associated to user; "bk,i", matrix associated to items; "pred", matrix containing the predicted scores.

```
#Matrix associated to users
rownames(output$au.k)<-c(paste("U",1:nrow(R),sep=""))
colnames(output$au.k)<-c(paste("K",1:k,sep=""))
print(output$au.k[1:10,])
```

```
##              K1           K2           K3           K4           K5
## U1  0.044019404 0.2237439650 0.4090046552 0.2673231710 0.0553874786
## U2  0.084450598 0.3123259240 0.0015268785 0.0004598964 0.2752945286
## U3  0.163729488 0.1507946256 0.0352598630 0.0801768776 0.2690710545
## U4  0.388584986 0.2106821465 0.0092735533 0.0009917525 0.0242156290
## U5  0.030006539 0.1974075432 0.0004912316 0.0004599939 0.4762735227
## U6  0.367101551 0.0975914578 0.0259843531 0.3533747010 0.0009928634
## U7  0.272939395 0.0008349031 0.2024366767 0.0411661362 0.2614093745
## U8  0.223436056 0.0425829032 0.2230252730 0.0382538448 0.1524720913
## U9  0.266041028 0.0555611635 0.0004795530 0.3689007756 0.0038922458
## U10 0.000461033 0.3928934331 0.0086327891 0.1603103357 0.1006302111
##               K6
## U1  0.0005213259
```

```
## U2  0.3259421748
## U3  0.3009680909
## U4  0.3662519327
## U5  0.2953611700
## U6  0.1549550736
## U7  0.2212135141
## U8  0.3202298318
## U9  0.3051252335
## U10 0.3370721980
```

```
#Matrix associated to items
colnames(output$bk.i)<-c(paste("I",1:ncol(R),sep=""))
rownames(output$bk.i)<-c(paste("K",1:k,sep=""))
print(output$bk.i[,1:7])
```

```
##           I1        I2        I3        I4        I5        I6        I7
## K1 0.6111342 0.5450023 0.5151848 0.4987901 0.5182718 0.5652517 0.5225188
## K2 0.5784046 0.5304977 0.5101690 0.4979192 0.5091319 0.5588532 0.5164537
## K3 0.5951232 0.5486916 0.5220282 0.4984588 0.5193577 0.5725022 0.5169789
## K4 0.6750241 0.5413172 0.5091344 0.5014438 0.5193805 0.5652845 0.5245977
## K5 0.6119211 0.5496087 0.5198883 0.5002061 0.5210142 0.5688044 0.5185094
## K6 0.6615516 0.5414260 0.5158864 0.4968971 0.5228324 0.5526160 0.5114281
```

```
#Predictions of the ratings
print(output$pred[1:10,1:7])
```

```
##           [,1]     [,2]     [,3]     [,4]     [,5]     [,6]     [,7]
##  [1,] 4.072058 3.712670 3.577525 3.496233 3.585608 3.834923 3.596119
##  [2,] 4.087833 3.702893 3.575747 3.491458 3.588304 3.800619 3.579491
##  [3,] 4.130712 3.714064 3.578307 3.493356 3.595656 3.808491 3.585794
##  [4,] 4.113190 3.703894 3.572782 3.489736 3.590471 3.797146 3.585163
##  [5,] 4.099804 3.716383 3.583213 3.493673 3.595609 3.809761 3.580672
##  [6,] 4.189572 3.709154 3.564242 3.496712 3.592546 3.814364 3.603975
##  [7,] 4.109272 3.729992 3.588508 3.493915 3.601277 3.824245 3.589882
##  [8,] 4.124394 3.723118 3.586039 3.491952 3.600229 3.815465 3.584708
##  [9,] 4.241319 3.708827 3.564549 3.495742 3.597879 3.805351 3.597732
## [10,] 4.127262 3.690013 3.565065 3.491875 3.583406 3.794520 3.581397
```

4. Performance of BNMF

We evaluate the performance of model through Mean Absolute Error (MAE). This measure calculates the accuracy of the predictios of this technique. The quality of the recommendations was evaluated with the metric Precision and Recall

```
#Processing the testing dataset
tmpRealRatingMatrixTst <- as(ratings.tst, "realRatingMatrix")
R.tst <- as(tmpRealRatingMatrixTst, "matrix")
R.tst[is.na(R.tst)] <- 0
R.tst[is.nan(R.tst)] <- 0
####Prediction Accuracy
mae(R.tst,output$pred)
```

```
## [1] 0.8077461
```

```
####Precision/Recall
vectPredictions <- c(5,10,20,40)
PrecisionRecall(output$pred,vectPredictions,ratings.tst)
```

```
##   TopN precision     recall         F1
## 1    5 0.7385230 0.12359701 0.21175528
## 2   10 0.7744154 0.07522715 0.13713311
## 3   20 0.7852684 0.04202298 0.07977678
## 4   40 0.8014019 0.02291555 0.04455703
```

Reference

Hernando, A., Bobadilla, J., & Ortega, F. (2016). A non negative matrix factorization for collaborative filtering recommender systems based on a Bayesian probabilistic model. Knowledge-Based Systems, 97, 188–202. http://doi.org/10.1016/j.knosys.2015.12.018