Reproducibility Report

Rayane Amrouche, Mamoune El Habbari, Amine Heffar, and Yassir Ramdani

EPITA, Kremlin-Bicêtre 94270, France

Abstract. In this report, we have taken upon ourselves to present the experiments we have accomplished to demonstrate the reproducibility of the paper called "Ordinal Non-negative Matrix Factorization for Recommendation" and written by Olivier Gouvert, Thomas Oberlin and Cédric Févotte. This paper introduces a new non-negative matrix factorization method that tackles ordinal data. We have done this and that and this and that.

Keywords: Matrix Factorization · Recommendation · Ordinal data.

1 The paper

Collaborative filtering has become a popular recommendation technique ever since Netflix put light onto it. This technique is based solely on users and their feedbacks on the items. This user/item interaction is modeled by a matrix Y of size $U \times I$ where U and I are the number of users and items respectively. The goal of these matrix factorization methods is to approximate the matrix Y by a low rank structure WH^T where W is a matrix that corresponds to the the users' preferences, meanwhile H corresponds to item attributes. While there are countless of different methods of matrix factorization, what this brings to the table is one that works on ordinal data.

1.1 Ordinal data

In an attempt to keep as much information as possible, the paper considers ordinal rather than binary data. What is ordinal data? It is a kind of nominal data that exhibits a natural ordering. It can be separated into two distinct categories: data with explicit or implicit feedback.

Ordinal data with explicit feedback has a clearly defined scale from which you can instantaneously determine if the item has received a positive feedback from the user or not. For example, when a movie is rated on a scale from 1 to 5, it is obvious that when a movie receives a 5-star rating, the user appreciated it.

On the other hand, implicit feedback comes into play when there is no real indicator of how positive or negative a feedback is. For example, if we look into songs play counts, we cannot actually say if a song is doing well based solely on these play counts. We need to rely on other parameters. If a song has been played 100000 times, we might want to say that it is a lot, but there may be other songs that have been played millions of times, or it might be the most played song in its specific sub-genre.

The challenge with ordinal data is that without a numerical value to rely on, we cannot apply the same algorithm that we might use on non-ordinal data and expect it to behave the same way. There are two naive ways to deal with ordinal data. The first one consists in applying classification methods which ignores the scale of the ordering relation between the different categories. The other one consists in treating the data as real values and applying basic regression models. The problem with these methods is that they have to drop crucial information. To overcome this problem, threshold methods have emerged. The goal of these methods is to train a predictive model which will learn the sequence of thresholds and thus mimic the regression models but with a more accurate distance between the classes.

1.2 Quantization

An important step in the processing of ordinal data is quantization. The objective is to quantize the non-negative real line $\mathbb{R}_+$ which basically means that every element of $\mathbb{R}_+$ is assigned an interval between thresholds that have been set beforehand. To this end, an increasing sequence of thresholds **b** is introduced alongside a quantization function which takes a value in $\mathbb{R}_+$ and returns which threshold interval it belongs to.

1.3 Generative model

Algorithms that attempt to learn the likelihood $p(y|x)$ directly or algorithms that attempt to learn the correspondences between the input and the labels are called discriminant models. In the context of recommender systems, they are often meant to infer the attractiveness of a given input to a user. More specifically, discriminant models of non-negative matrix factorization are meant to infer W and H so that $W * H^T = Y$ with Y being the relation matrix between users and items.

Conversely, algorithms that attempt to learn the posterior $p(x|y)$ are generative. For example in the binary case, if y indicates whether a movie is attractive to a specific user, then $p(x|y = 1)$ models the distribution of the relevant characteristics of the movie, and $p(x|y = 0)$ models the distribution of the irrelevant characteristics of the movie to the customer. In the context of ordinal data the posterior is way more complex. Indeed y is not binary anymore but continuous or in our case ordinal. Thus, if y indicates the rating of a movie for a specific user, $p(x|y = r)$ models the distribution of characteristics rated "r" of the movie.

The main axis of development in this paper is thus the new probabilistic framework based on previous work of the same author [1]. This paper described a binary generative model called DcPF. Thus, OrdNMF try to offer better results on ordinal data. Notwithstanding the obvious possibilities a generative model can offer, this kind of model is difficult to figure out. Indeed, instead of affixing a simple probabilistic consideration, generative models are based on Bayesian theory. The main idea is to obtain the posterior from a biased prior, likelihood and observation made on the data.

1.4 Bayesian inference

The problem when computing the posterior is that it is simply intractable because of the computation of the distribution of the observation that leads to an integral computation of probabilities. The variational inference offers a way to approach this distribution by a simpler variational distribution through optimization. Each variational distribution is multiplied because we consider each of them as independent to the others. This is a trick in variational inference that allows an easier approach to the original distribution with a bias.

Variational inference consists in minimizing the kullback-Leibler divergence. This is a metric used to compare two variational distributions. A computation trick allows us to reduce the computation impact of the distribution of the observation that is needed in the process of optimization using KL divergence. This trick allows us to simply compute ELBO, the lower bound of the evidence (the evidence being the distribution of the observation).

However, ELBO also implies that we should compute an expectation and thus a distribution integral. Thus, the updates rules are still intractable. However, now we can now simply use empirical expectation that we have through observation.

The optimization algorithm proposed by the paper is CAVI for Coordinate Ascent Variational Inference. The authors also stated that perhaps a different algorithm would have been better. This algorithm works by iteratively optimizing each of the latent variables conditioned on the rest of the latent variables.

1.5 Model Augmentation

The aim of the model is thus to infer W and H (as well as the thresholds for the quantization). Multiple model augmentation tricks are proposed in the paper to simplify the variational inference update rules of the latent factors W and H, as well as those of the thresholds b. These tricks mainly aim to get prior and likelihood to conjugate to infer more easily. However some tricks also allow the model to take efficiently advantage of the sparsity of Y.

The paper decided to use a multiplicative inverse-gamma and thus gamma distribution as a prior on W and H for our variational inference model. It appears that BePoF, an other generative model that is based on a Poisson distribution and that has been augmented to handle binarized data, is a special case of a IG-OrdNMF.

That being said, the authors used the same model augmentation on IG-OrdNMF that were used on BePoF. This simplified the computation of the update rules ans also allowed to benefit from the sparsity of the relation matrix. An other model augmentation was added and is based on the PF approach. This model augmentation is very common and simplify even more the update rules. It consists in a multinomial distribution that would suppress some of the complicated computation added by the previous distribution variable added.

2 Experiences

2.1 Code & Python

A particularity of the paper we tried to reproduce is that a github repository has been provided to be able to test the results. The different scripts and models have been coded in Python 2.7.

The code has several particularities that make it strange to run. For some reason even using the right version of python, we were unable to run a lot of functions and scripts. After some corrections we have been able to reproduce some of the results by using the same seeds.

However Python 2 is quite strange and outdated so actually a lot of the results are not interpretable and even glitched. Thus, we decided to port the code to Python 3. Some functions used are classic functions of collaborative filtering so it was quite easy to understand what was their utility and thus it was easy to port. Nevertheless, some functions we had to port were actually difficult to understand. This lead to a lot of code review and a lot of time was consumed to correct some of these functions.

We first focused on data preprocessing. Some parameters of pandas functions have completely changed their behavior between Python 2 and Python 3 which lead to a lot of confusion for us and a lot of time was consumed before getting the exact same results as the authors. In the end the dataset obtained with the same seed as the authors is identical to the one we obtained.

Then we ported the models. We focused on OrdNMF because we wanted to focus on the main subject of the paper. Apart from some imports that have changed the code was completely usable even in python 3 but we spent a lot of time to compare all the formulas with the one of the original paper. Some of the functions were not explicitly explained in the paper and thus we also checked the thesis of the authors [2] that had a lot more information and we can confirm that the code provided is the same as the one described in the thesis.

Finally we ported the different tests and evaluation scripts. Once again the different small changes between Python 2 and Python 3 were corrected and quickly most of the code was usable in Python 3. We were now able to completely retest all the evaluation shown in the paper and even test the model with other datasets thanks to the work we have done to understand the preprocessing used by the authors.

2.2 Prediction

While OrdNMF is a generative model, it is still first supposed to infer W and H. In addition, OrdNMF is also a threshold model and thus a key of the prediction is the learning of the thresholds b that allow us to handle ordinal data from continuous prediction.

It aims to maximize the mathematical expectation of the log of the distribution of Y and Z given the thresholds b with respect to b (with Z being the product of W, H and all the augmented variables added for the mean of the inference). To simplify, training the threshold consist of maximising the likelyhood of the threshold and thus of finding the best distribution of the data given the thresholds.

In there experiments, the authors compared OrdNMF with DcPF but also PF and BePoF. The two last are very similar to OrdNMF but they are not thresholds models so the data need to be binarized.

Two datasets are used: Movie Lens (ML) and Taste Profile (TP) dataset. The TP dataset contains continuous data, thus, DcPF can be used with raw data. In the other hand, OrdNMF has been created for ordinal data. So the TP dataset has been quantized. In the case of ML, the data is already ordinal so OrdNMF can be used without any preprocessing. However the authors also binarized the dataset to be able to use BePoF and PF models.

Tests have been realized with the exact same seed on each portion of the code provided by the authors to ensure that the results are the same. After running the models with the exact same seeds the results are the same. Because we have carefully read the code we can now make sure that the results really correspond to the ones presented in the paper.

2.3 Posterior Predictive Check

Generative models can turn out to be very powerfull in the context of recommender system. Indeed, not only a generative model can infer W and H so that $W * H^T = Y$, such a model can also predict interaction between item and users that are not known yet.

In there experiments, the authors used posterior predictive checks (PPCs) to demonstrate the flexibility of OrdNMF and its ability to represent various kinds of datasets (explicit and implicit ordinal data). It basically consist of generating new data based on the posterior predictive distribution obtained through training on known data.

By comparing original data and predicted data obtained thanks to the posterior distribution approximation, the authors try to show the effectiveness of OrdNMF to describe the ordinal classes. The results shown are unilateral : OrdNMF succeeded to describe all the ordinal classes while DcPF have some difficulties with large values.

Once again, we reproduced the same text from the code we ported to Python 3, and once again the results are unanimous: we obtained the exact same graph. The conclusion of our code porting and test reproduction is that all the results shown in the paper are obtained from the code provided. Because we have read the code and ported it we can also confirm that there is no false reproduction of the formulas in the code. From now on the debate is to know if then the results obtained in the paper have been biased by selecting specific seeds.

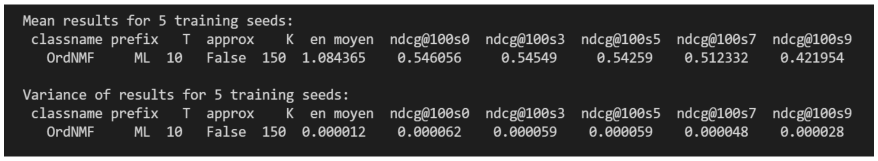
2.4 Synthetic dataset experiments

To experiment the model and make tests on several seeds, in order to verify that the results are not the fruits of chance, we decided to reproduce them on a synthetic dataset.

For that, we took MovieLens 100k instead of 20M and we restarted the training of the OrdNMF model on 5 different seeds. To verify the accuracy and to validate the effectiveness of the model we have done several tests and we have computed the variance of the results on the paper metrics.

As we can see in this image, the NDCG score shows that it performs better on the 100k dataset than in the 20M one.

These results are the mean on the 5 different seeds, and as we can see the variance between these different experiments is around 10e-5. This seems to indicate that the results of the paper do not seem to change and can be reliable.



```
Mean results for 5 training seeds:
 classname prefix   T  approx    K  en moyen  ndcg@100s0  ndcg@100s3  ndcg@100s5  ndcg@100s7  ndcg@100s9
   OrdNMF     ML  10   False  150  1.084365    0.546056     0.54549     0.54259    0.512332    0.421954

Variance of results for 5 training seeds:
 classname prefix   T  approx    K  en moyen  ndcg@100s0  ndcg@100s3  ndcg@100s5  ndcg@100s7  ndcg@100s9
   OrdNMF     ML  10   False  150  0.000012    0.000062    0.000059    0.000059    0.000048    0.000028
```

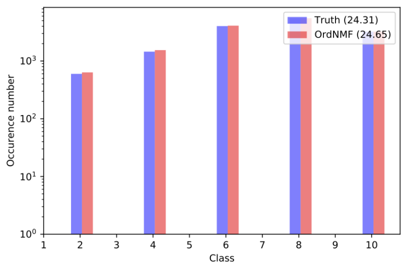
Fig. 1: Recommendation performance of OrdNMF using the MovieLens 100k dataset, evaluated on NDCG score. With Threshold s

We also plotted the PPC of the distribution of the classes in the MovieLens dataset (100k). The blue bars (Truth) represent the histogram of the classes in the train set. The colored bars represent the simulated histograms obtained from the different inferred OrdNMF models. The percentages of non-zero values are written in parentheses.

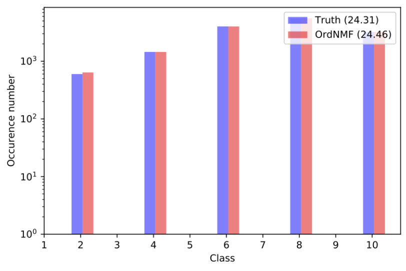
One thing to note is that on the 100k dataset, ratings are between 0 and 5 and when we plotted the PPC of the classes distribution PPC graph, we got only 5 bars, whereas in the 20m dataset we got 10 bars. This is due to the ratings which fluctuate between 1 and 10 and we got 10 classes.

These graphs show how well the OrdNMF model describes the ordinal categories, and even with several training seeds of the model, the results are consistent overs different trainings.

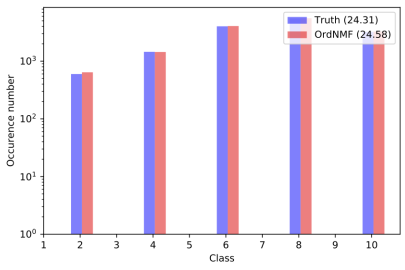
We also computed the variance on the sparsity coefficient, and we obtained a variance of 0.02 for the 5 training seeds, which seems to confirm the reliability of this model.



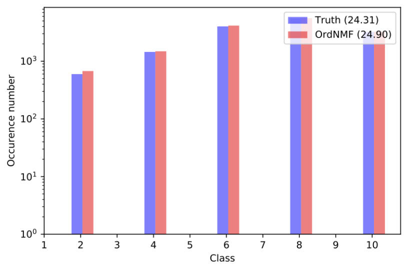
(a) seed = 0



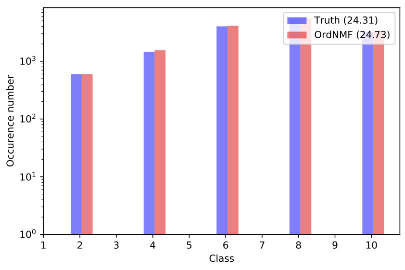
(b) seed = 1



(c) seed = 2



(d) seed = 3



(e) seed = 4

References

1. O. Gouvert, T. Oberlin, C. Févotte ”Recommendation from Raw Data with Adaptive Compound Poisson Factorization”
2. O. Gouvert, T. Oberlin, C. Févotte ”Factorisation bayésienne de matrices pour le filtrage collaboratif, Thèse de doctorat Informatique et Télécommunications Toulouse, INPT 2019”