ESANN 2018 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 25-27 April 2018, i6doc.com publ., ISBN 978-287587047-6.
Available from http://www.i6doc.com/en/.

# Neural Networks for Implicit Feedback Datasets

Josef Feigl and Martin Bogdan

University of Leipzig - Department of Computer Engineering
Augustusplatz 10, 04109 Leipzig - Germany

**Abstract**. Most users typically interact with products only through implicit feedback such as clicks or purchases rather than explicit user-provided information like product ratings. Learning to rank products according to individual preferences using only this implicit feedback can be helpful to make useful recommendations. In this paper, a neural network architecture to solve collaborative filtering problems for personalized rankings on implicit feedback datasets is presented. It is shown how a layer of constant weights forces the network to learn pairwise rankings. Additionally, similarities between the network and a matrix factorization model trained with Bayesian Personalized Ranking are proven. The experiments indicate state-of-the-art performance for the task of personalized ranking.

## 1 Introduction and Related Work

Matrix factorization methods like Singular Value Decomposition are very common in recommender systems on explicit feedback datasets but can also be applied on implicit feedback datasets [1]. However, these methods are usually not optimized to learn personalized item rankings. A popular method to overcome this problem is Bayesian Personalized Ranking ($BPR$), which attempts to directly learn user-specific preferences between two items [2]. In this paper, we propose a neural network architecture to learn pairwise preferences and show similarities between our model and a matrix factorization model trained with $BPR$.

This paper is structured as followed: A brief description of the general problem is given in section 2. Afterwards in section 3, we explain the neural network architecture, derive all weight updates and show similarities between our model and a matrix factorization model trained with $BPR$. The proposed model is evaluated in section 4. We summarize our findings in section 5.

## 2 Preliminaries

Let $U = \{1, \dots, N\}$ be a set of users and $I = \{1, \dots, M\}$ a set of items with $N, M \in \mathbb{N}$. We have a dataset of observed interactions $S \subseteq U \times I$, where $(u, i) \in S$ means that user $u$ interacted with item $i$ in some way. Each observation $(u, i) \in S$ is regarded as positive feedback and $(u, i) \in (U \times I) \setminus S$ as negative feedback. We use $I_u^+$ as a short notation for the set of all positive items of user $u \in U$ and $I_u^-$ for the set of all negative items. The measure of preference of user $u$ for item $i$ is given by $x_{ui} \in \mathbb{R}$: A user $u$ prefers item $i$ over item $j$ if $x_{ui} > x_{uj}$. We denote $\hat{x}_{ui}$ as the prediction made by a model for $x_{ui}$ and define $x_{uij} \coloneqq x_{ui} - x_{uj}$ as the difference between the preferences of the positive and negative item.

ESANN 2018 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 25-27 April 2018, i6doc.com publ., ISBN 978-287587047-6.
Available from http://www.i6doc.com/en/.

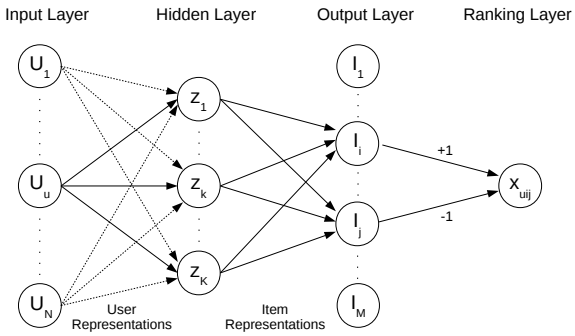


Fig. 1: Architecture of the neural network: The solid lines connecting to the hidden layer represent the weights of user $u$. The lines connecting to the upper item unit represent the weights for the positive item unit $i$. The lower connected item unit represents the negative item unit $j$.

To achieve the task of learning personalized item rankings for each user, our goal is to maximize the probability that for a user $u$ the relationship $\hat{x}_{ui} > \hat{x}_{uj}$ holds true if $i \in I_u^+$ and $j \in I_u^-$. This can be formalized measuring the area under the ROC curve ($AUC$):

$$AUC := \frac{1}{|U||I_u^+||I_u^-|} \sum_{u \in U} \sum_{i \in I_u^+} \sum_{j \in I_u^-} H(\hat{x}_{uij}), \tag{1}$$

where $H$ is the Heaviside function [2]. The score for a perfect model is 1, whereas a model with random predictions would achieve a score of 0.5.

# 3 Model

## 3.1 Overview and Notation

Our proposed model is a modified feedforward neural net with four specific layers $L$: An user layer $L^1$ with $N$ units, a hidden layer $L^2$ with $K$ units, an item layer $L^3$ with $M$ units and a ranking layer $L^4$ with one unit (see figure 1). Therefore, the user layer $L^1$ has as many units as there are users and the item layer $L^3$ has as many units as there are items. The parameter $K$ determines the size of the user and item representations [3].

The following short notations are used in this paper: Let $\mathbf{W}^l$ be the set of all weights connecting to the $l^{th}$ layer. We define $\mathbf{W}^l_{i\cdot}$ as the set of all weights connecting from layer $L^{l-1}$ to unit $i$ of layer $L^l$ and $\mathbf{W}^l_{\cdot j}$ as the set of all incoming weights from the $j^{th}$ unit of layer $L^{l-1}$ to all units of layer $L^l$. Additionally, $\mathbf{a}^l$ defines the activation or output of layer $l$.

Following this notation, $\mathbf{W}^2$ can be interpreted as the representations of the users $U$ and thus, $\mathbf{W}^2_{\cdot u}$ as the representation of user $u \in U$. Analogically, we

ESANN 2018 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 25-27 April 2018, i6doc.com publ., ISBN 978-287587047-6. Available from http://www.i6doc.com/en/.

interpret $\mathbf{W}^3$ as the representations of the items $I$ and $\mathbf{W}^3_{i\cdot}$ as the latent factor of item $i \in I$. The weights $\mathbf{W}^2 \in \mathbb{R}^{N\times K}$ and $\mathbf{W}^3 \in \mathbb{R}^{K\times M}$ will be interpreted as weight matrices for the rest of this paper. Both are initialized with uniformly distributed random numbers from the range $[-0.01, 0.01]$.

### 3.2 Forward-Propagation

Let $R = \{(u, i, j) \mid u \in U, i \in I_u^+, j \in I_u^-\}$ be a set of training examples. We will show the forward propagation for a single training triple $(u, i, j) \in R$.

A binarized version $\mathbf{a}^1 = 1_u \in \{0, 1\}^N$ of $u$ is used as the input for the network. It is defined as the indicator vector $1_u \coloneqq (x_0, x_1, \cdots, x_N)$ with $x_j = 1$ if $j = u$ and $x_j = 0$ otherwise. Using $1_u$ as input for the network implies that only the weights $\mathbf{W}^2_{\cdot u}$ contribute anything to the input of the hidden layer $L^2$. The output $\mathbf{a}^2 \in \mathbb{R}^K$ is given by

$$\begin{aligned} \mathbf{a}^2 &= f(\mathbf{W}^2\mathbf{a}^1) \\ &= f(\mathbf{W}^2_{\cdot u}), \end{aligned} \tag{2}$$

where $f : \mathbb{R} \to \mathbb{R}$ is the activation function [3].

It is sufficient for our model to only compute the output for the units of the positive item $i$ and the negative item $j$. Therefore, only the weights $\mathbf{W}^3_{i\cdot}$ and $\mathbf{W}^3_{j\cdot}$ are used to compute the output $\mathbf{a}^3$, which is given by

$$\hat{x}_{uz} \coloneqq a^3_z = f(\mathbf{W}^3_{z\cdot}\mathbf{a}^2) \tag{3}$$

for $z = \{i, j\}$.

The ranking layer consists of only one unit and two incoming constant weights with $W^4_{1i} = 1$ and $W^4_{1j} = -1$. Thus, the positive weight is always connected to the positive item unit of the previous layer and vice versa for the negative weight. The output of this layer is therefore given by:

$$\mathbf{a}^4 = \sigma(a^3_i - a^3_j) = \sigma(\hat{x}_{uij}). \tag{4}$$

We are using the logistic sigmoid $\sigma$ as the activation function for this layer to get the probability estimate that user $u$ prefers item $i$ over item $j$.

The output of the neural net is determined by the user weights $\mathbf{W}^2_{\cdot u}$ and the item weights $\mathbf{W}^3_{i\cdot}$ and $\mathbf{W}^3_{j\cdot}$. We are calling these weights *active* and denote $\mathbf{W} \coloneqq \mathbf{W}(u, i, j)$ as the set of all *active* weights for the training triple $(u, i, j)$.

### 3.3 Backpropagation

To better show similarities between our model and $BPR$, a simplified version of our proposed model, which uses only identity activation functions in both the hidden and the item layer, is used for this section.

The cross entropy $C$ is used as the cost function for the network:

$$C = -\frac{1}{|R|}\sum_{(u,i,j)\in R} y \cdot \ln \sigma(\hat{x}_{uij}) + (1 - y) \cdot \ln\left(1 - \sigma(\hat{x}_{uij})\right). \tag{5}$$

ESANN 2018 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 25-27 April 2018, i6doc.com publ., ISBN 978-287587047-6.
Available from http://www.i6doc.com/en/.

The training set $R$ is missing target values $y$ in the classical machine learning sense, but since it is our goal to maximize the probability $\sigma(\hat{x}_{uij})$, we can set $y = 1$ for every training sample and thus:

$$C = -\frac{1}{|R|} \sum_{(u,i,j) \in R} \ln \sigma(\hat{x}_{uij}). \tag{6}$$

The general update rule to find the weight updates for all active weights $\mathbf{W}$ is given by

$$\mathbf{W} \leftarrow \mathbf{W} + \alpha \Delta \mathbf{W}, \tag{7}$$

where $\alpha$ is the learning rate. To derive the updates $\Delta \mathbf{W}$ for every active weight, the error $\delta$ of the ranking layer is propagated backwards through the net [4]. Let $\delta_i^l$ be the error for unit $i$ in layer $l$. Furthermore, let $\Delta \mathbf{W}_{i\cdot}^l$ be the update for weight $\mathbf{W}_{i\cdot}^l$. The error for the ranking layer $\delta_1^4$ is given by:

$$\begin{aligned} \delta_1^4 &= y - \sigma(\hat{x}_{uij}) \\ &= \sigma(-\hat{x}_{uij}). \end{aligned} \tag{8}$$

As stated above, the incoming weights of the ranking layer are constant and thus not updated. Backpropagating the error $\delta_1^4$ to the positive and the negative item units of the item layers yields

$$\delta_i^3 = \sigma(-\hat{x}_{uij}), \tag{9}$$

$$\delta_j^3 = -\sigma(-\hat{x}_{uij}), \tag{10}$$

and results in the following weight updates:

$$\Delta \mathbf{W}_{i\cdot}^3 = \mathbf{W}_{\cdot u}^2 \cdot \sigma(-\hat{x}_{uij}), \tag{11}$$

$$\Delta \mathbf{W}_{j\cdot}^3 = -\mathbf{W}_{\cdot u}^2 \cdot \sigma(-\hat{x}_{uij}). \tag{12}$$

Finally, we need to backpropagate the error to the active unit of the user layer and update the weights of user $u$. The error $\delta_u^2$ is given by:

$$\begin{aligned} \delta_u^2 &= \mathbf{W}_{i\cdot}^3 \cdot \sigma(-\hat{x}_{uij}) - \mathbf{W}_{j\cdot}^3 \cdot \sigma(-\hat{x}_{uij}) \\ &= (\mathbf{W}_{i\cdot}^3 - \mathbf{W}_{j\cdot}^3) \cdot \sigma(-\hat{x}_{uij}). \end{aligned} \tag{13}$$

We are only updating the active unit for user $u$ in the user layer and since $1_u$ is used as input to the net, the update for the weights of the user $u$ is given by:

$$\Delta \mathbf{W}_u^2 = \delta_u^2. \tag{14}$$

#### *3.3.1 Connections to Bayesian Personalized Ranking*

The network proposed in this paper is similar to a matrix factorization model since one can interpret the user and item weights $W^2$ and $W^3$ as the two latent factor matrices resulting from a decomposition of the target matrix [3].

ESANN 2018 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 25-27 April 2018, i6doc.com publ., ISBN 978-287587047-6. Available from http://www.i6doc.com/en/.

The optimization criterion $BPR$ was proposed by Rendle et al. in [2]. They also applied it to a matrix factorization model and derived all weight updates for such a model, which can be found in [2, Section 4.3.1]. By comparing them, one can see that these update rules as well as the output of the model are equal to those of the simplified version of our network (see (11), (12) and (14)).

### 3.4 Mini-Batch-Processing

Instead of processing a single sample at a time, a whole mini-batch of $p$ samples is processed at once: to create a mini-batch of size $p$, we choose a uniformly randomly selected set of $p$ users $u \in U$. For each user, one of his positive and one of his negative items is randomly selected (uniformly distributed with replacement). One mini-batch is processed in every training iteration.

## 4 Experiments and Results

### 4.1 Setting

The MovieLens 1M dataset and the Netflix Prize dataset are used to evaluate our model. Since both datasets contain explicit ratings, they are converted to implicit ones by only keeping samples with a rating above 3. A random sample of 10 000 users is used for the Netflix Prize dataset. Each dataset is split randomly into two sets of equal size. These sets will be our train and test set. We made sure that all users and movies in the test set occur at least once in the training set. This way, the sets created from the MovieLens 1M and the Netflix Prize datasets consist of about 270 000 and 450 000 user-movie-samples, respectively. The described process is repeated three times and the results for each model are averaged.

Our model is compared against two popular baselines model: A weighted matrix factorization model ($WRMF$ [1]) and a matrix factorization model using the $BPR$ optimization criterion ($BPRMF$ [2]). The publicly available software $MyMediaLite$ was used to compute the results for the baseline models [5].

The $AUC$ metric as given by (1) and the $Precision@5$ metric, which gives the ratio of relevant items in the top 5 recommended items, are used to measure the performance of all models.

Our network uses $SELU$ activation functions in the item layer, $L2$ Regularization for all active weights and the Adam optimizer for all weight updates [6, 7]. We use a hidden layer size of $K = 100$ and hold out 10% of the training set as a validation set to optimize all other model parameters.

### 4.2 Results

The evaluation of all models can be found in Table 1. The matrix factorization models trained using the $BPR$ criterion achieved a superior $AUC$ performance compared to the $WRMF$ model in all experiments. This is no surprise, since these models are directly optimizing this metric. Our model achieved a significantly stronger predictive performance on the MovieLens 1M dataset for both

ESANN 2018 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 25-27 April 2018, i6doc.com publ., ISBN 978-287587047-6.
Available from http://www.i6doc.com/en/.

Table 1: Comparison with baseline models

| | MovieLens 1M | | Netflix Prize | |
|---|---|---|---|---|
| | *AUC* | *Precision*@5 | *AUC* | *Precision*@5 |
| *WRMF* | 0.9160 | 0.4579 | 0.9302 | **0.3847** |
| *BPRMF* | 0.9339 | 0.4814 | 0.9529 | 0.3742 |
| *Our Model* | **0.9370** | **0.4949** | **0.9546** | 0.3844 |

metrics compared to both benchmark models. On the Netflix Prize dataset, it performed about as good as the $WRMF$ model for the $Precision@5$ metric and as the $BPR$ model for the $AUC$ metric.

## 5 Summary

In this paper, we have designed a neural network architecture to achieve a matrix factorization which directly optimizes the $AUC$ metric to learn personalized item rankings. It was shown how to train this network with the $BPR$ optimization criterion using a layer of constant weights and the common cross entropy cost function. We have proven that a simplified version of our network is similar to a matrix factorization model trained with $BPR$. Due to the transition to neural networks we were able to easily integrate modern advancements of this domain into our model and thus further improve the predictive performance. The evaluation of our model using two datasets indicates a state-of-the-art performance of this approach.

## References

[1] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, ICDM '08, pages 263–272, 2008.

[2] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, UAI '09, pages 452–461, Arlington, Virginia, United States, 2009. AUAI Press.

[3] Josef Feigl and Martin Bogdan. Collaborative filtering with neural networks. In *ESANN 2017, 25th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, pages 441–446, 2017.

[4] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Neurocomputing: Foundations of research. chapter Learning Representations by Back-propagating Errors, pages 696–699. MIT Press, Cambridge, MA, USA, 1988.

[5] Zeno Gantner, Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. MyMediaLite: A free recommender system library. In *Proceedings of the 5th ACM Conference on Recommender Systems (RecSys 2011)*, 2011.

[6] Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. *CoRR*, abs/1706.02515, 2017.

[7] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.