Dual Neural Personalized Ranking

Seunghyeon Kim
Sungkyunkwan University
South Korea
kimsh1509@skku.edu

Jongwuk Lee*
Sungkyunkwan University
South Korea
jongwuklee@skku.edu

Hyunjung Shim
Yonsei University
South Korea
kateshim@yonsei.ac.kr

ABSTRACT

Implicit user feedback is a fundamental dataset for personalized recommendation models. Because of its inherent characteristics of sparse one-class values, it is challenging to uncover meaningful user/item representations. In this paper, we propose *dual neural personalized ranking* (*DualNPR*), which fully exploits both user- and item-side pairwise rankings in a unified manner. The key novelties of the proposed model are three-fold: (1) DualNPR discovers mutual correlation among users and items by utilizing both user- and item-side pairwise rankings, alleviating the data sparsity problem. We stress that, unlike existing models that require extra information, DualNPR naturally augments both user- and item-side pairwise rankings from a user-item interaction matrix. (2) DualNPR is built upon deep matrix factorization to capture the variability of user/item representations. In particular, it chooses raw user/item vectors as an input and learns latent user/item representations effectively. (3) DualNPR employs a dynamic negative sampling method using an exponential function, further improving the accuracy of top-N recommendation. In experimental results over three benchmark datasets, DualNPR outperforms baseline models by 21.9–86.7% in hit rate, 14.5–105.8% in normalized discounted cumulative gain, and 5.1–23.3% in the area under the ROC curve.

CCS CONCEPTS

• **Information systems → Collaborative filtering**.

KEYWORDS

Implicit feedback; top-N recommendation; dual representation

ACM Reference Format:
Seunghyeon Kim, Jongwuk Lee, and Hyunjung Shim. 2019. Dual Neural Personalized Ranking. In *Proceedings of the 2019 World Wide Web Conference (WWW '19), May 13–17, 2019, San Francisco, CA, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3308558.3313585

1 INTRODUCTION

With the enormous success of e-commerce and online streaming service (*e.g.*, Amazon, eBay, Spotify, and Netflix), recommendation systems have gained much attention in academia and industry [24, 27, 35]. As one of the most successful techniques, collaborative filtering (CF) has been widely used for building a personalized recommendation model. The underlying goal of CF is to predict user preferences on unrated items by discovering hidden correlations among users/items. Concretely, CF is formulated to address either *rating prediction* or *top-N recommendation.* While rating prediction minimizes the error of predicting a user's ratings for her unrated items, top-N recommendation finds a ranked list of the favorite top-N items among unrated items.

Depending on the inherent characteristics of datasets, the recommendation models exploit *explicit* or *implicit* feedback. Earlier approaches have employed user's explicit feedback (*e.g.*, 5-star ratings). While explicit feedback is useful to understand relative user preferences among rated items, it is not always available in practice. By contrast, implicit feedback (*e.g.*, purchase histories, bookmarks, and browsing logs) does not require a user to annotate ratings but is naturally obtained from the user's behavior. Therefore, it is much easier to collect than explicit feedback, and thereby it is more feasible for real-world applications. In this paper, we aim to develop a new recommendation model for the top-N recommendation from implicit feedback.

Existing recommendation models with implicit feedback commonly suffer from the following challenges: (1) Implicit feedback is extremely sparse (*i.e., data sparsity*). In most real datasets, the sparsity is about 96–99%, which leads to it being difficult for the recommendation models to learn meaningful latent user/item representations. (2) The latent user/item representations are usually induced from simple matrix factorization, which assumes that a user-item interaction matrix is represented by a linear combination of latent user vectors/item vectors. When handling highly complicated real-life datasets, this linear model might be too simple to disentangle the latent user/item vectors. (3) While implicit feedback only provides positive data for labling positive experiences, the correct label of unknown feedback is ambiguous (*i.e., data ambiguity*). Unknown feedback is interpreted as a mixture of negative and unlabeled positive feedback. Therefore, it is necessary to distinguish negative feedback from unknown feedback.

To overcome these challenges, various recommendation models have been proposed in the past decade [24]. Among them, Bayesian personalized ranking (BPR) [23] is one of the most representative recommendation models. Unlike previous approaches [10, 14, 19, 25] that directly predict user preferences for unrated items, BPR minimizes a pairwise ranking loss between positive and negative feedback. In this way, it adequately addresses the data sparsity problem. Because BPR still relies on simple matrix factorization, it is incapable of correctly disentangling complex latent user/item representations. In addition, it does not pay much attention to how to distinguish negative feedback from unknowns; BPR regards all unknowns as negative feedback.

Recent studies attempt to inherit the advantage of BPR at handling data sparsity while improving the modeling power. Several

*Corresponding author

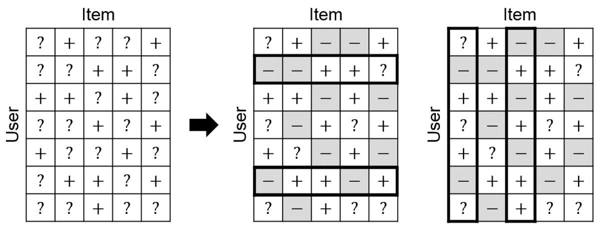


Figure 1: Example of user- and item-side pairwise rankings (center and right) elicited from a user-item interaction matrix (left), where gray color indicates negative feedback from unknown feedback. For instance, user 2 prefers item 3 to item 1 (center) and item 1 is preferred by user 3 over user 6 (right).

studies [4, 6, 18] have integrated the BPR model with deep neural networks (DNNs) to provide more expressiveness (*i.e.*, additional parameters) and flexibility (*i.e.*, non-linear activation functions) in discovering hidden correlations among users/items. Most recently, neural personalized ranking (NPR) [18] has been proposed to generalize the BPR model using DNNs. However, it focused solely on incorporating auxiliary information into DNNs and has not yet studied deep matrix factorization (DMF) to learn latent user/item representations as well as negative sampling to choose negative items from unknown feedback.

In this paper, we propose *dual neural personalized ranking* (*DualNPR*), which incorporates both user- and item-side pairwise rankings into DMF. (Figure 3 depicts the detailed architecture of the proposed model.) The key novelties of the proposed model are summarized as follows.

(1) **Utilizing dual pairwise rankings**: To alleviate the data sparsity problem, we introduce a novel data representation to *fully* exploit both user- and item-side pairwise rankings in a unified manner. Figure 1 illustrates the user- and item-side pairwise rankings collected from a toy user-item interaction matrix. While original BPR only utilizes user-side pairwise ranking, it is intuitive to employ both user- and item-side pairwise rankings to better understand correlations among users/items; it is particularly beneficial to investigate their *mutual* correlations. The use of item-side pairwise ranking is also attractive because we can augment pairwise rankings without requiring any side information. To the best of our knowledge, none of the existing work utilizes both user- and item-side pairwise rankings. Moreover, the idea of adding item-side pairwise ranking can be easily extended to improve the performance of BPR and its variants such as BPR with heterogeneous implicit feedback [20] and NPR [18].

(2) **Incorporating BPR into DNNs**: Motivated by DMF [34], our model adopts multiple hidden layers in DNNs to increase the model capacity, and learns more meaningful latent user/item representations. We believe that our model architecture helps capture the variability of user/item representations and mutual correlations among users/items. In particular, our model takes *raw* user/item vectors as an input. Although this idea is simple, user/item vectors with similar patterns are enforced to be placed as close as possible in latent space. Moreover, we can leverage a denoising method [16, 33, 36] for raw input vectors to make our model more generalized.

(3) **Dynamic negative sampling**: We design *dynamic negative sampling* using an exponential function to distinguish negative items from unknown feedback. Without much computational overhead, our negative sampling dynamically maximizes the gradient gap between positive and negative feedback in training our model. Consequently, it helps improve the quality of our model as well as achieve stable learning convergence.

We conduct extensive experiments with public benchmark datasets including Amazon and MovieLens datasets. Experimental results show that the proposed model significantly outperforms competitive models by 21.9–86.7% in terms of hit rate (HR), 14.5–105.8% in normalized discounted cumulative gain (NDCG), and 5.1–23.3% in the area under the ROC curve (AUC). Furthermore, our model shows an improvement of 35.3–43.3% in HR when both using user- and item-side triplets. In addition, our dynamic negative sampling outperforms existing sampling methods by 3.3–4.6% in HR.

2 PRELIMINARIES

In this section, we first introduce basic notations and formulate the top-N recommendation problem. Then, we review the most relevant existing work such as BPR [23] and NPR [18].

Problem Statement. Let $\mathcal{U}$ be a set of m users, and $\mathcal{I}$ be a set of n items. We are given a user-item interaction matrix $\mathbf{R} \in \{0,1\}^{m \times n}$, where $r_{ui} \in \mathbf{R}$ indicates implicit feedback from user $u \in \mathcal{U}$ for item $i \in \mathcal{I}$. Specifically, $r_{ui} = 1$ means that user feedback is *observed* (or *known*), and it is interpreted as positive feedback. Meanwhile, $r_{ui} = 0$ indicates *unobserved* (or *unknown*) feedback, and it is interpreted as a mixture of negative and unlabeled positive feedback. Given user u, $\mathcal{I}_u^+ = \{i \in \mathcal{I} | r_{ui} = 1\}$ is defined as a set of items with positive feedback by user u. Likewise, $\mathcal{U}_i^+ = \{u \in \mathcal{U} | r_{ui} = 1\}$ is defined as a set of users with positive feedback for item i. For unobserved feedback, $\mathcal{I}_u^- = \mathcal{I} \backslash \mathcal{I}_u^+$ and $\mathcal{U}_i^- = \mathcal{U} \backslash \mathcal{U}_i^+$ denote a set of items and a set of users, respectively.

In this paper, we aim to optimize a top-N recommendation list using pair-wise user/item preferences (*i.e.*, a pair of positive and negative users/items). To formalize this problem, a set D_u of *user-side triplets* is defined as follows:

$$\mathcal{D}_u = \{(u,i,j) | u \in \mathcal{U} \wedge i \in \mathcal{I}_u^+ \wedge j \in \mathcal{I}_u^-\}, \tag{1}$$

where (u,i,j) represents that user u prefers i to j. The pairwise relationship is denoted as $i >_u j$. Likewise, a set of *item-side triplets* is defined as follows:

$$\mathcal{D}_i = \{(i,u,v) | i \in \mathcal{I} \wedge u \in \mathcal{U}_i^+ \wedge v \in \mathcal{U}_i^-\}, \tag{2}$$

where (i,u,v) means that item i is preferred by user u over user v, denoted as $u >_i v$. Although these relationships are useful resources for learning recommendation models, existing studies [4, 6, 18, 23] built upon BPR solely leverage user-side triplets from a user-item interaction matrix.

Bayesian Personalized Ranking (BPR). Because implicit user feedback only provides sparse one-class values for positive feedback, it is inherently difficult to uncover the relative preferences of users. This is also known as the one-class collaborative filtering (OCCF) problem [10, 19]. To address this problem, BPR [23] is formulated to minimize a pairwise ranking loss such that all items with positive feedback from a target user are ranked higher than the rest of the items.

Specifically, BPR aims to maximize the posterior probability $p(\Theta | i >_u j)$, where Θ is a set of learning parameters. According to Bayes' rule, the posterior probability is defined as:

$$p(\Theta | i >_u j) \propto p(i >_u j | \Theta) p(\Theta). \tag{3}$$

Then, the likelihood function for Θ is defined as:

$$p(i >_u j | \Theta) = \sigma(\hat{r}_{ui} - \hat{r}_{uj}), \tag{4}$$

where $\sigma(\cdot)$ is the sigmoid function and $\hat{r}_{ui}$ and $\hat{r}_{uj}$ are the predicted scores for items i and j by user u, respectively. In BPR, $\hat{r}_{ui}$ is computed by a linear combination of latent user vectors and item vectors [10, 14, 19, 25].

$$\hat{r}_{ui} = \mathbf{p_u}^T \mathbf{q_i}, \tag{5}$$

where $\mathbf{p_u} \in \mathbb{R}^{d \times 1}$ is a latent user vector and $\mathbf{q_i} \in \mathbb{R}^{d \times 1}$ is a latent item vector. (Symbols in bold font are vectors and matrices.) In other words, the model parameters of BPR are $\Theta = \{\mathbf{P}, \mathbf{Q}\}$; $\mathbf{P}$ and $\mathbf{Q}$ indicate a set of latent user vectors and a set of latent item vectors, respectively. The prior density follows a normal distribution $p(\Theta) \sim \mathcal{N}(0, \lambda_\Theta \mathbf{I})$, where λ_Θ is a model-specific parameter and $\mathbf{I}$ is the identity matrix.

Because of the monotone transformation property of the natural log function, maximum likelihood estimation is equivalent to minimizing the negative log likelihood. Finally, based on Bayesian inference, the objective function of BPR aims to minimize a pairwise ranking loss for all user-side triplets.

$$\underset{\Theta}{\text{argmin}} \sum_{(u,i,j) \in \mathcal{D}_u} - \ln \left(\sigma(\hat{r}_{ui} - \hat{r}_{uj}) \right) + \Omega(\Theta), \tag{6}$$

where Ω is Frobenius regularization for parameters Θ.

Neural Bayesian Personalized Ranking (NPR). To benefit from the great success of DNNs [15], there several extensions of BPR using DNNs. First, visual BPR [6] combines the BPR model with auxiliary visual features learned from convolutional neural networks (CNN). Also, Bayesian personalized ranking deep neural network [4] introduces CNN to infer latent user representations from social graphs. In other words, DNNs have been exploited as a feature extractor for additional contexts and inputs for graphs, but have not been extended to BPR to learn complex user/item representations.

Most recently, NPR [18] is proposed to generalize BPR using DNNs. Figure 2 illustrates the architecture of NPR by integrating BPR with DNN. Specifically, it takes a user-side triplet (u, i, j) as an input, where user u and a pair of items i and j are represented by *one-hot* vectors. For instance, in Figure 1, user 2 prefers item 3 to item 1. In this case, $\mathbf{u}$, $\mathbf{i}$, and $\mathbf{j}$ are represented by $[0, 1, 0, 0, 0, 0, 0]$, $[0, 0, 1, 0, 0]$, and $[1, 0, 0, 0, 0]$, respectively. Because there are m users and n items in $\mathbf{R}$, one-hot input vectors $\mathbf{u}$, $\mathbf{i}$, and $\mathbf{j}$ are represented by m-, n-, and n-dimensional vectors, respectively, and the three embedding vectors are computed by:

$$\mathbf{p_u} = \mathbf{Wu}, \quad \mathbf{q_i} = \mathbf{W'i}, \quad \mathbf{q_j} = \mathbf{W''j}, \tag{7}$$

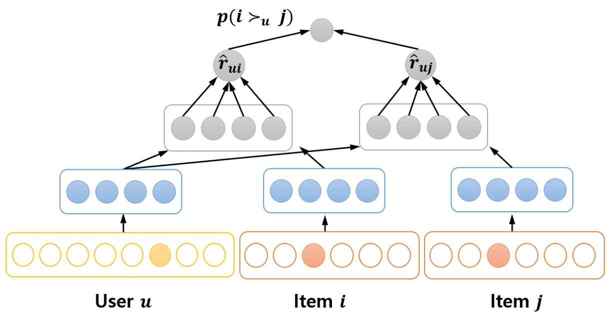


Figure 2: Model architecture of neural personalized ranking (NPR). Note that the user/item is represented by an one-hot vector.

where $\mathbf{W}$, $\mathbf{W'}$, and $\mathbf{W''}$ are embedding matrices for user u and items i and j, respectively. Because they transform the input vector to latent user/items vectors, they can correspond to learning parameters $\mathbf{P}$ and $\mathbf{Q}$ in Eq. (5).

As NPR is vertically symmetric for a user-item pair, we mainly explain the substructure for user u and item i in Figure 2. Unlike existing simple matrix factorization, $\hat{r}_{ui}$ is calculated by a non-linear matrix factorization model.

$$\hat{r}_{ui} = a(\mathbf{w}^\mathbf{T}(\mathbf{p_u} \odot \mathbf{q_i}) + b), \tag{8}$$

where $\odot$ is an element-wise product and $\mathbf{w} \in \mathbb{R}^{d \times 1}$ is a d-dimensional weight vector for latent dimensions and b is the bias term. In addition, $a(\cdot)$ indicates the activation function. In [18], the ReLU function is used as the activation function to capture non-linear user-item interactions. It is reported that Eq. (8) not only identifies the variability of user-item interactions by relaxing an equal weight requirement but also offers non-linear flexibility.

Except for computing $\hat{r}_{ui}$ and $\hat{r}_{uj}$ in Eq. (8), the objective function of NPR is very similar to Eq. (6) in original BPR. It is expected that non-linear matrix factorization used in NPR is effective for representing user/item vectors in a more flexible manner. As a result, NPR can achieve a better quality for top-N recommendation than BPR. However, it does not address the following issues: (1) item-side triplets are also a useful training resource, (2) more generalized DMF can be incorporated to represent latent user/item vectors, and (3) negative sampling is one of the key components for learning BPR and its variants.

3 PROPOSED MODEL

In this section, we propose *dual neural personalized ranking* (*DualNPR*), which complements the drawbacks of BPR and its variants. The key technical contributions of DualNPR can be summarized as follows: (1) We adopt a novel data representation to unify both user- and item-side triplets. Such a unified representation can be utilized for all pairs in user- and item-side rankings. Because the representation can also capture mutual correlations among users/items, we can effectively alleviate the data sparsity problem. (2) Our model is built upon a DMF model with multiple hidden layers to obtain the variability of user/item representations. In this process, raw user/item vectors are taken as an input. The input vectors encode

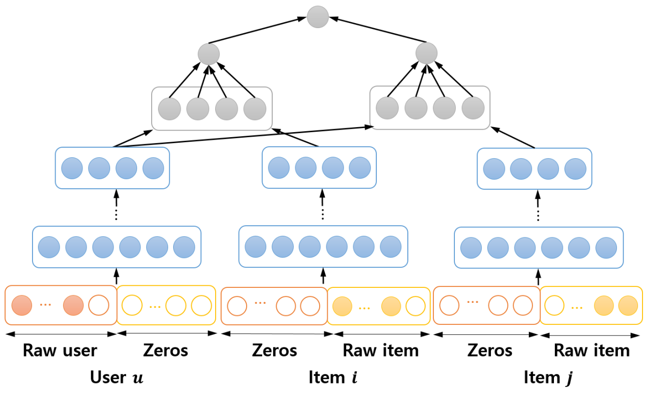


Figure 3: Model architecture of DualNPR. The user/item is represented by a raw vector concatenated with a zero vector.

richer information than conventional one-hot vector representations. Additionally, it is advantageous to apply a denoising scheme for input vectors. (3) We employ a new dynamic negative sampling strategy to choose negative feedback from unknown feedback effectively. This sampling method helps to improve the quality of our model as well as achieve stable learning convergence. In short, DualNPR can be interpreted as a more generalized BPR model built upon DNN without any additional information.

3.1 Model Architecture

Figure 3 illustrates the network architecture of DualNPR, integrating both user- and item-side triplets. Specifically, it consists of the following components: *input layer*, *embedding layer*, *scoring layer*, and *output layer*.

Input Layer: As depicted in Figure 3, while the NPR model uses a one-hot vector representation for user u, our model defines each user u by a *raw* vector across all items. Similarly, each item i is represented by a raw vector across all users. In Figure 1, for instance, when $user_2$ prefers $item_3$ to $item_1$, $\mathbf{u}$, $\mathbf{i}$, and $\mathbf{j}$ are represented by $[0,0,1,1,0]$, $[0,1,0,1,0,1,1]$, and $[0,0,1,0,1,0,0]$, respectively. While the similarity between two one-hot vectors is difficult to quantify, the similarity between two raw vectors is intuitive and able to be measured by distance metrics. That is, we can expect that the distance between latent vectors should have a similar tendency to the distance between corresponding raw vectors; the similar raw vectors result in similar latent vectors. Moreover, we can apply a denoising scheme for the input layer to increase the robustness of the proposed model as in existing work [16, 33, 36]. Formally, each input vector for a user-side triplet (u, i, j) is represented by:

$$\mathbf{u} = \mathbf{R}_{u*}, \quad \mathbf{i} = \mathbf{R}_{*i}, \quad \mathbf{j} = \mathbf{R}_{*j}, \tag{9}$$

where $\mathbf{R}_{u*} \in \{0,1\}^{n\times 1}$, $\mathbf{R}_{*i} \in \{0,1\}^{m\times 1}$, and $\mathbf{R}_{*j} \in \{0,1\}^{m\times 1}$ are a row vector for user u and column vectors for items i and j in a user-item matrix $\mathbf{R}$, respectively.

We now explain a *dual* data representation to exploit both user and item vectors in a unified manner. To represent user/item vectors, existing recommendation models [7, 8, 18, 34, 36] using DNN usually define user/item representation by different layers with user/item embedding matrices. In contrast to existing models, we adopt a single, unified input vector representation, which is defined by the following $(m+n)$-dimensional concatenated vector.

$$[\mathbf{u}; \mathbf{i}] \in \{0,1\}^{m+n}. \tag{10}$$

To represent user vector $\mathbf{u}$ with the concatenated vector, the elements for the item vector are set as zeros. Likewise, to represent item vector $\mathbf{i}$, the elements for the user vector are set as zeros. Formally:

$$\mathbf{u} = [\mathbf{R}_{u*}; \mathbf{0}], \quad \mathbf{i} = [\mathbf{0}; \mathbf{R}_{*i}], \quad \mathbf{j} = [\mathbf{0}; \mathbf{R}_{*j}]. \tag{11}$$

This dual representation is capable of sharing one embedding layer instead of using two embedding layers for users/items. We believe that such unified embedding is a key factor to better formulate the personalized ranking problem because it learns mutual correlations across users/items.

Embedding Layer: To learn latent user/item vectors, we use one or multiple hidden layers as used in DMF [34]. For a user-side triplet (u, i, j), the first layer is represented by:

$$l_1 = a(\mathbf{W_1} x + b_1) \ \text{ for } \ x \in \{\mathbf{u}, \mathbf{i}, \mathbf{j}\}, \tag{12}$$

where $\mathbf{W_1} \in \mathbb{R}^{(m+n)\times d}$ is an embedding matrix for users/items, b_1 is the bias term, and $a(\cdot)$ is the activation function. Given (u, i, j), we have three input vectors, $\mathbf{u}$, $\mathbf{i}$ and $\mathbf{j}$. Each is a $(m+n)$-dimensional vector and all three vectors share the same embedding layer. We can also employ multiple intermediate hidden layers.

$$l_i = a(\mathbf{W_i} l_{i-1} + b_i) \ \text{ for } \ i = 2, \ldots, L. \tag{13}$$

Through the multiple hidden layers, (u, i, j) is represented by three low-dimensional vectors in a latent space. We denote the three embedding vectors as $\Phi(\mathbf{u})$, $\Phi(\mathbf{i})$, and $\Phi(\mathbf{j})$.

$$\Phi(\mathbf{u}) = a(\ldots a(\mathbf{W_2} a(\mathbf{W_1}[\mathbf{R}_{u*}; \mathbf{0}] + b_1) + b_2) \ldots), \tag{14}$$

$$\Phi(\mathbf{i}) = a(\ldots a(\mathbf{W_2} a(\mathbf{W_1}[\mathbf{0}; \mathbf{R}_{*i}] + b_1) + b_2) \ldots), \tag{15}$$

$$\Phi(\mathbf{j}) = a(\ldots a(\mathbf{W_2} a(\mathbf{W_1}[\mathbf{0}; \mathbf{R}_{*j}] + b_1) + b_2) \ldots), \tag{16}$$

where $a(\cdot)$ is used as the ReLU function at each layer. From Eq (7) in NPR, $\Phi(\mathbf{u})$, $\Phi(\mathbf{i})$, and $\Phi(\mathbf{j})$ correspond to $\mathbf{p_u}$, $\mathbf{q_i}$, and $\mathbf{q_j}$, respectively, indicating a latent user vector and two latent item vectors.

The key advantage of sharing one embedding layer for both users and items is to being able to leverage an item-side triplet (i, u, v) in a unified manner. That is, one embedding layer through concatenated input vectors can be used for both user- and item-side triplets. As a result, the embedding layer can model mutual information among users/items, which effectively increases the modeling capacity with a smaller number of parameters in comparison with NPR.

Scoring Layer: Similar to traditional latent factor models [10, 14, 19], the scoring layer is used to predict a preference score for user-item interactions. Formally,

$$\hat{r}_{ui} = g\Big(\Phi(\mathbf{u}), \Phi(\mathbf{i})\Big), \tag{17}$$

where $g(\cdot,\cdot)$ is the scoring function for a user-item pair. In general, this function can be represented by a (non-linear) inner product and cosine similarity. In this paper, we investigate the following scoring functions:

- **Inner product**: As the simplest method, this is computed by the inner product between two vectors.

$$\hat{r}_{ui} = \Phi(\mathbf{u})^T \Phi(\mathbf{i}). \tag{18}$$

- **Cosine similarity**: This is computed by the normalized inner product. This function is also used for a deep structured semantic model (DSSM) [11] and a DMF model [34].

$$\hat{r}_{ui} = \frac{\Phi(\mathbf{u})^T \Phi(\mathbf{i})}{||\Phi(\mathbf{u})||\ ||\Phi(\mathbf{i})||}. \tag{19}$$

- **Non-linear inner product**: As in NPR [18], this is computed with a non-linear weighted inner product by appending weights for latent dimensions.

$$\hat{r}_{ui} = a(\mathbf{w}^T(\Phi(\mathbf{u}) \odot \Phi(\mathbf{i})) + b). \tag{20}$$

One might ask whether we should consider more complex scoring functions to improve the modeling power. One of the representative recommendation models using DNNs, neural collaborative filtering [8] successfully employs a complex scoring function using multi-layer perception. Such a deeper network can be useful for capturing non-linear and complex relationships between latent user/item vectors. However, in the current form of our model, we have kept relatively simple scoring functions. This is because we can employ multiple hidden layers for learning latent user/item vectors. Having a complex scoring model may deteriorate the overall quality of the proposed model by incurring the overfitting problem of DNN. Therefore, we skip using complex scoring functions. (In future work, we will study how to optimize the scoring function.)

Output Layer: Given two predicted scores $\hat{r}_{ui}$ and $\hat{r}_{uj}$, we minimize a pairwise ranking loss for Θ. For user- and item-side triplets, the likelihood functions are defined as:

$$p(i >_u j|\Theta) = \sigma(\hat{r}_{ui} - \hat{r}_{uj}) = \sigma\Big(g\big(\Phi(\mathbf{u}), \Phi(\mathbf{i})\big) - g\big(\Phi(\mathbf{u}), \Phi(\mathbf{j})\big)\Big), \tag{21}$$

$$p(u >_i v|\Theta) = \sigma(\hat{r}_{iu} - \hat{r}_{iv}) = \sigma\Big(g\big(\Phi(\mathbf{i}), \Phi(\mathbf{u})\big) - g\big(\Phi(\mathbf{i}), \Phi(\mathbf{v})\big)\Big). \tag{22}$$

Objective Function: Finally, we introduce the objective function to deal with both user- and item-side pairwise rankings. Similar to classical BPR, our objective function aims to minimize the relative loss for pairwise rankings. In clear contrast, our objective function is to minimize both user- and item-side ranking losses.

$$\begin{aligned} \underset{\Theta}{\text{argmin}} & \sum_{(u,i,j)\in\mathcal{D}_u} -\ln\left(\sigma(\hat{r}_{ui} - \hat{r}_{uj})\right) + \\ & \lambda \sum_{(i,u,v)\in\mathcal{D}_i} -\ln\left(\sigma(\hat{r}_{iu} - \hat{r}_{iv})\right) + \Omega(\Theta), \end{aligned} \tag{23}$$

where $\sigma(\hat{r}_{ui} - \hat{r}_{uj})$ and $\sigma(\hat{r}_{iu} - \hat{r}_{iv})$ are computed by Eq. (21) and Eq. (22), respectively, and λ is the weight to control the importance of item-side triplets. $\Omega(\Theta)$ is the Frobenius regularization term for our parameters $\Theta = \{\mathbf{W}_1, \ldots, \mathbf{W}_L, b_1, \ldots, b_L\}$.

3.2 Dynamic Negative Sampling

One of the main components in BPR is to choose negative items from unknown feedback. The original BPR model [23] is based on a *uniform* negative sampling method. Given user u, it first determines the number of negative items. In general, the number of negative items is proportional to the number of positive items for user u, *i.e.*, $|\mathcal{I}_u^-| = \alpha \cdot |\mathcal{I}_u^+|$. (Usually, α is set to 2–5.) Then, negative items are randomly selected from a set of all unrated items with an *equal* probability. Although this is straightforward, the uniform sampling method is ineffective for improving the quality of BPR.

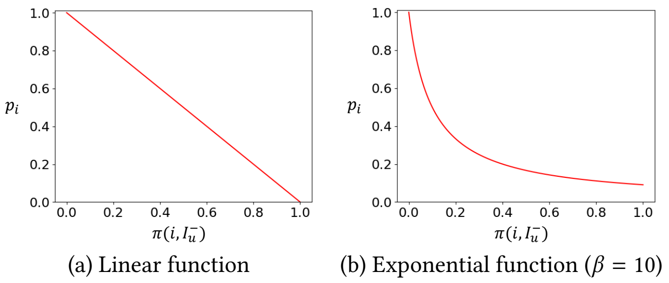


Figure 4: Comparison of linear and exponential functions in the Amazon Music dataset.

To address this problem, a dynamic negative sampling strategy [22, 37] is proposed to maximize *gradient gaps* between positive and negative items. Intuitively, it chooses the negative items that have a substantial prediction loss at each learning step. For the current model, the probability of choosing a negative item is based on a ranked list of all unrated items. That is, the importance $\pi(i, \mathcal{I}_u^-)$ of negative item $i \in \mathcal{I}_u^-$ for dynamic negative sampling is derived by the ranking of negative items.

$$\pi(i, \mathcal{I}_u^-) = \frac{rank(i)}{|\mathcal{I}_u^-|}, \tag{24}$$

where $rank(i)$ is the ranking order of item i in $\mathcal{I}_u^-$. If item i has the highest ranking in $\mathcal{I}_u^-$, $\pi(i, \mathcal{I}_u^-)$ is $1/|\mathcal{I}_u^-|$. That is, when an item is ranked highest, $\pi(i, \mathcal{I}_u^-)$ is lowest. Then, the probability p_i of selecting negative item $i \in \mathcal{I}_u^-$ is computed by a *linear* function.

$$p_i \propto 1 - \pi(i, \mathcal{I}_u^-). \tag{25}$$

We modify this dynamic negative sampling by replacing the linear function with an exponential function. Figure 4 illustrates the linear and exponential function for dynamic negative sampling. Under this adaption, the items with high rankings have a much higher probability than negative items. The probability of selecting item i is rewritten as follows:

$$p_i \propto \frac{1}{\exp(\beta\pi(i, \mathcal{I}_u^-))}, \tag{26}$$

where β is the parameter used to control the slope of the exponential function. Note that our dynamic sampling can also be performed for both user- and item-side triplets.

3.3 Training and Prediction

Algorithm 1 describes the detailed procedure for training the proposed model. To update the parameters on each layer, we use back-propagation by adopting mini-batch gradient descent. For each epoch, we shuffle a training dataset for the user- and item-side triplets. In Section 4.1, we will explain the detailed parameter settings for training our model.

Algorithm 1: Training procedure of DualNPR.

```
Input: A user-item matrix $\mathbf{R}$, the number of epochs $T$
Parameters: $\Theta = \{\mathbf{W}_1, \ldots, \mathbf{W}_L, b_1, \ldots, b_L\}$
1  Randomly initialize parameters $\Theta$.
2  for $t \leftarrow 1$ to $T$ do
3      for $u \in \mathcal{U}$ and $i \in \mathcal{I}$ do
4          Set negative samples from $\mathcal{I}_u^-$ and $\mathcal{U}_i^-$ using Eq. (26).
5      end
6      Randomly sample a mini-batch $\mathcal{S} \subseteq \mathcal{D}_u$ and $\mathcal{S}' \subseteq \mathcal{D}_i$.
7      for $(u, i, j) \in \mathcal{S}$ do
8          Compute $\sigma(\hat{r}_{ui} - \hat{r}_{uj})$ using Eq. (21).
9      end
10     for $(i, u, v) \in \mathcal{S}'$ do
11         Compute $\sigma(\hat{r}_{iu} - \hat{r}_{iv})$ using Eq. (22).
12     end
13     Update $\Theta$ with the pairwise ranking error in Eq. (23).
14 end
```

After training our model, we need to calculate the scores for all unrated items $\mathcal{I}_u^-$ for user u for top-N recommendation. Given user u and every unrated item $i \in \mathcal{I}_u^-$, the user's preference score $\hat{r}_{ui}$ is computed from the proposed model. In this case, we simply feed a user-item pair (u, i) to a substructure of the proposed model. Because our model shares the same learning parameters for a triplet input, we can simply obtain predicted score $\hat{r}_{ui}$ by using the substructure. After all unrated items for user u are passed through our model, they are sorted by descending order. Finally, we select the top-N items with the highest scores.

3.4 Extensions to BPR Variants

We explain how to extend our dual representation to several BPR variants. First, BPR with confidence (BPRC) [31] adds a confidence weight for each implicit feedback on top of the BPR framework. The objective function is defined as:

$$\underset{\Theta}{\operatorname{argmin}} \sum_{(u,i,j) \in \mathcal{D}_u} -\ln\left(c_{uij}\sigma(\hat{r}_{ui} - \hat{r}_{uj})\right) + \Omega(\Theta), \tag{27}$$

where c_{uij} is a confidence weight for (u, i, j). Our idea of dual representation can be extended to BPRC; adding the confidence for user- and item-side triplets can effectively achieve such an extension. Let c_{uij} and c_{iuv} denote confidence weights for (u, i, j) and (i, u, v), respectively. The objective of BPRC with our dual representation is formulated as:

$$\begin{aligned} \underset{\Theta}{\operatorname{argmin}} & \sum_{(u,i,j) \in \mathcal{D}_u} -\ln\left(c_{uij}\sigma(\hat{r}_{ui} - \hat{r}_{uj})\right) + \\ \lambda & \sum_{(i,u,v) \in \mathcal{D}_i} -\ln\left(c_{iuv}\sigma(\hat{r}_{iu} - \hat{r}_{iv})\right) + \Omega(\Theta). \end{aligned} \tag{28}$$

We further extend our model to adaptive BPR (ABPR) [20] with different implicit feedback. The objective of ABPR with our dual representation is written as follows:

$$\begin{aligned} \underset{\Theta}{\operatorname{argmin}} & \sum_{(u,i,j) \in \mathcal{D}_u} -\ln\left(c_{uij}\sigma(\hat{r}_{ui} - \hat{r}_{uj})\right) + \\ \lambda_{\mathcal{E}} & \sum_{(u,i,j) \in \mathcal{E}_u} -\ln\left(c'_{uij}\sigma(\hat{r}_{iu} - \hat{r}_{iv})\right) + \Omega(\Theta), \end{aligned} \tag{29}$$

where $\mathcal{E}_u$ is a set of user-side triplets collected from another implicit feedback dataset, and $\lambda_{\mathcal{E}}$ is the regularization term for $\mathcal{E}_u$. Analogous to Eq. (28), we simply add item-side triplets to two different implicit feedback datasets in Eq. (29).

Lastly, we discuss how to apply our dual representation to NPR and contextual NPR. For NPR, we can replace input vectors in Eq. (7) with concatenated user-item vectors. Then, their input vectors are modified by:

$$\mathbf{p_u} = \mathbf{W}[\mathbf{u}; \mathbf{0}], \quad \mathbf{q_i} = \mathbf{W}[\mathbf{0}; \mathbf{i}], \quad \mathbf{q_j} = \mathbf{W}[\mathbf{0}; \mathbf{j}], \tag{30}$$

where $\mathbf{W}$ indicates one embedding matrix for users/items. (In Section 4, we will evaluate the extension of NPR by adding our dual representation.)

Similarly, our dual representation can also be extended to contextual NPR. Let $\mathbf{e}_u^c$ and $\mathbf{e}_i^c$ denote a user context vector and an item context vector, respectively. For input vectors in Eq. (10), we append two contextual vectors to a user-item input vector, *i.e.*, $[\mathbf{u}; \mathbf{e}_u^c; \mathbf{i}; \mathbf{e}_i^c]$. Then, a user vector is represented by setting an item vector and its context as zeros. Likewise, an item vector is represented by setting a user vector and its context as zeros.

$$\mathbf{u} = [\mathbf{R}_{u*}; \mathbf{e}_u^c; \mathbf{0}; \mathbf{0}], \quad \mathbf{i} = [\mathbf{0}; \mathbf{0}; \mathbf{R}_{*i}; \mathbf{e}_i^c], \quad \mathbf{j} = [\mathbf{0}; \mathbf{0}; \mathbf{R}_{*j}; \mathbf{e}_j^c] \tag{31}$$

Under this dual representation, we can also share one embedding layer for users/items with contexts.

Table 1: Statistics of three real-world datasets.

Statistics	AMusic	MLLatest	ML1M
# of users	2,831	671	6,040
# of items	13,410	9,066	3,706
# of ratings	63,064	100,004	1,000,208
Sparsity (%)	99.83%	98.36%	95.53%

4 EXPERIMENTS

In this section, we conduct extensive experiments to evaluate the effectiveness of the proposed model. First, we describe the data preparation and experimental setup. Then, we compare DualNPR with several competitive models, including the state-of-the-art recommendation model. Also, we carry out an ablation study to investigate the effect of dual representation, that of negative sampling, that of scoring functions, and finally that of DMF.

4.1 Experimental Setup

Datasets. We perform extensive experiments over three real-world datasets including Amazon Music (AMusic), MovieLens Latest (ML-Latest) and MovieLens 1M (ML1M). These are all publicly available

Table 2: Performance comparisons between DualNPR and other baseline models (best in bold).

Datasets	Models	Measures						
		HR@25	HR@50	HR@100	NDCG@25	NDCG@50	NDCG@100	AUC
AMusic	BPR	0.0797	0.1200	0.1754	0.0311	0.0342	0.0477	0.6866
	NPR	0.0454	0.0684	0.1111	0.0163	0.0207	0.0276	0.7118
	Dual+NPR	0.0678	0.1053	0.1590	0.0254	0.0326	0.0413	0.8000
	DualNPR	**0.1488**	**0.2136**	**0.2992**	**0.0580**	**0.0704**	**0.0843**	**0.8780**
	Gain (%)	+86.7	+78.0	+70.6	+86.5	+105.8	+76.7	+23.3
MLLatest	BPR	0.1770	0.2248	0.2820	0.0798	0.0889	0.0982	0.7325
	NPR	0.1463	0.2054	0.2879	0.0572	0.0731	0.0819	0.8365
	Dual+NPR	0.1580	0.2221	0.2683	0.0579	0.0703	0.0779	0.6639
	DualNPR	**0.2158**	**0.2903**	**0.3851**	**0.0914**	**0.1056**	**0.1210**	**0.9244**
	Gain (%)	+21.9	+29.1	+33.8	+14.5	+18.8	+23.2	+10.5
ML1M	BPR	0.1488	0.2181	0.3166	0.0589	0.0723	0.0882	0.8370
	NPR	0.1454	0.2181	0.3172	0.0567	0.0706	0.0866	0.8672
	Dual+NPR	0.1454	0.2197	0.3113	0.0583	0.0726	0.0873	0.8240
	DualNPR	**0.2292**	**0.3290**	**0.4556**	**0.0994**	**0.1185**	**0.1390**	**0.9115**
	Gain (%)	+54.0	+49.7	+43.6	+68.8	+63.2	+57.6	+5.1

on their websites [1] [2]. Because we are interested in the recommendation models with implicit feedback, all of the observed feedback was regarded as positive feedback. For Movielens datasets, we do not apply any preprocessing. For AMusic, because it is notorious for high sparsity, we filtered out unreliable ratings; users who had less than 10 ratings and items that had been rated by less than five users were eliminated as has been done in existing studies [8, 33, 34]. Table 1 reports the detailed statistics of the three datasets.

Competitive Models. We compared DualNPR with two existing models and NPR with our dual representation.

- **BPR** [23]: This model minimizes a pairwise ranking loss for user-side triplets using simple matrix factorization.
- **NPR** [18]: This is a state-of-the-art BPR variant that combines BPR with DNNs.
- **Dual+NPR**: Using our dual representation, we extend NPR to utilize both user- and item-side triplets.

Although several models [7, 8, 34, 36] utilize DNNs for personalized ranking, these models belong to the point-wise approach, which directly predicts the absolute rating. In this paper, we choose BPR and NPR as the competitive models because both of them aim to optimize a pairwise ranking loss. Also, because some BPR variants [4, 6] have exploited DNNs as a feature extractor for additional contexts and inputs for graphs, it is inappropriate to compare our model with them.

Evaluation Protocol. To assess the accuracy of top-N recommendation, we adopted the *leave-one-out evaluation*, which has been widely used in the literature [7, 8, 34]. We held-out one random user-item interaction as the test item for each user, and the rest of user-item interactions were used for the training dataset. Unlike sampling-based evaluation tasks [7, 8, 34] that randomly choose 100 items that have not been rated by the user, we choose all items unrated by each user as the candidate items for top-N recommendation. Because there are many unrated items for each user, our evaluation protocol should handle much more challenging cases for top-N recommendation. We believe that our evaluation protocol is more suitable for conducting a fair evaluation for top-N recommendation because it essentially prevents sampling bias during evaluation.

Evaluation Measures. For each model, we reported the accuracy of top-N recommendation with three metrics: hit rate (HR), normalized discounted cumulative gain (NDCG) and area under the ROC curve (AUC). These metrics have been widely used in existing studies [7, 8, 23, 34, 36]. The size of the ranked list N is chosen as 25, 50, and 100. While the HR@N intuitively measures whether a test item is present in the top-N list, the NDCG@N puts more weight on highly ranked hit items than others in the top-N list. Lastly, AUC accounts for the relative rank of the test item among all unrated items. Essentially, AUC is optimized by BPR [23]. Note that all three measures are averaged across all users. The higher they are, the more accurate a recommendation model is.

Implementation Details. We implemented DualNPR and other existing models using TensorFlow. We randomly initialized model parameters with Xavier initializer [5]. For all baseline models, the number of latent dimensions was set to 128. The optimizer for the baseline models was mini-batch Adagrad. We optimized the models by adjusting the batch size of 32 and the learning rate of 0.005. The regularization term for the learning parameters was set to 0.0001. For DualNPR, the weight λ of user-item triplets was set to 0.6 (AMusic) and 0.8 (MLLatest and ML1M). All experiments were performed on a desktop with 128 GB memory and 2 Intel Xeon Processors (E5-2630 v4, 2.20 GHz, 25M cache), and all models were trained using 4 Nvidia GeForce GTX 1080Ti.

[1]https://grouplens.org/datasets/movielens
[2] http://jmcauley.ucsd.edu/data/amazon/

Table 3: Effects of dual representation in DualNPR (best in bold).

Datasets	Dual rep.	Measures						
		HR@25	HR@50	HR@100	NDCG@25	NDCG@50	NDCG@100	AUC
AMusic	No	0.0783	0.1227	0.1856	0.0291	0.0376	0.0478	0.8103
	Yes	**0.1122**	**0.1707**	**0.2511**	**0.0451**	**0.0561**	**0.0691**	**0.8706**
	Gain (%)	+43.3	+39.1	+35.3	+55.0	+49.2	+44.6	+7.4
MLLatest	No	0.1431	0.2066	0.2918	0.0571	0.0692	0.0829	0.8964
	Yes	**0.2158**	**0.2903**	**0.3851**	**0.0914**	**0.1056**	**0.1210**	**0.9244**
	Gain (%)	+50.8	+40.5	+32.0	+60.1	+52.6	+46.0	+3.1
ML1M	No	0.2090	0.3034	0.4236	0.0859	0.1042	0.1234	0.9042
	Yes	**0.2292**	**0.3290**	**0.4556**	**0.0994**	**0.1185**	**0.1390**	**0.9115**
	Gain (%)	+9.7	+8.4	+7.6	+15.7	+13.7	+12.6	+0.8

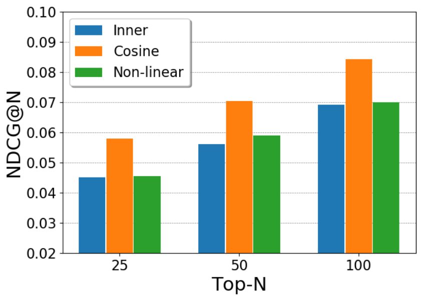
Table 4: Effects of negative sampling in DualNPR (best in bold).

Datasets	Functions	Measures						
		HR@25	HR@50	HR@100	NDCG@25	NDCG@50	NDCG@100	AUC
AMusic	Random	0.0988	0.1495	0.2239	0.0389	0.0486	0.0606	0.8535
	Linear	0.1441	0.2043	0.2894	0.0567	0.0683	0.0821	0.8756
	Exponential	**0.1488**	**0.2136**	**0.2992**	**0.0580**	**0.0704**	**0.0843**	**0.8780**
	Gain (%)	+3.3	+4.6	+3.4	+2.3	+3.1	+2.7	+0.3
MLLatest	Random	0.1878	0.2626	0.3589	0.0747	0.0891	0.1046	0.9090
	Linear	0.2137	0.2858	0.3802	**0.0941**	**0.1079**	**0.1232**	0.9217
	Exponential	**0.2158**	**0.2903**	**0.3851**	0.0914	0.1056	0.1210	**0.9244**
	Gain (%)	+1.0	+1.6	+1.3	-2.9	-2.1	-1.8	+0.3
ML1M	Random	0.1612	0.2427	0.3501	0.0641	0.0798	0.0971	0.8813
	Linear	0.2173	0.3110	0.4319	0.0915	0.1098	0.1294	0.9040
	Exponential	**0.2292**	**0.3290**	**0.4556**	**0.0994**	**0.1185**	**0.1390**	**0.9115**
	Gain (%)	+5.5	+5.8	+5.5	+8.6	+7.9	+7.4	+0.8

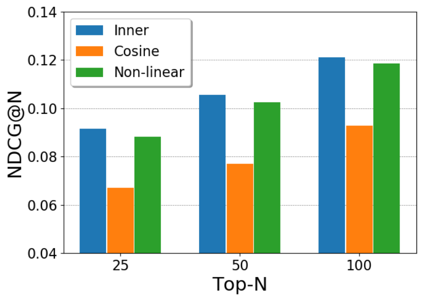
4.2 Experimental Results

DualNPR vs. Other Competitors. We evaluate the effectiveness of DualNPR in comparison with two baseline models and NPR with dual representation. In DualNPR, by default, while we used cosine similarity as the similarity function for AMusic, we used inner product as the similarity function for MLLatest and ML1M. As shown in Table 2, we found three key observations: (1) Regardless of datasets and measures, DualNPR significantly outperformed the baseline models. DualNPR was consistently better than the baseline models by 21.9–86.7% in HR, 14.5–105.8% in NDCG, and 5.1–23.3% in AUC. For the experimental results of NPR and BPR, our results show a similar tendency as reported in [18]. This achievement is meaningful in the sense that we have not added any auxiliary information. (2) Modifying NPR with our dual representation is beneficial at improving the accuracy of the original NPR. In AMusic, we observed that Dual+NPR was improved by 43.1–49.3% in HR and 49.6–57.4% in NDCG, and 12.3% in AUC, respectively. Although Dual+NPR is less accurate than DualNPR, it significantly outperformed NPR. (3) Lastly, DualNPR consistently outperforms Dual+NPR: 36.5–57.6% in HR, 50.2–70.5% in NDCG, and 10.6–39.2% in AUC. This improvement comes from DMF and negative sampling. Therefore, we can conclude that our choice of DMF and negative sampling is effective to improve the accuracy of top-N recommendation.

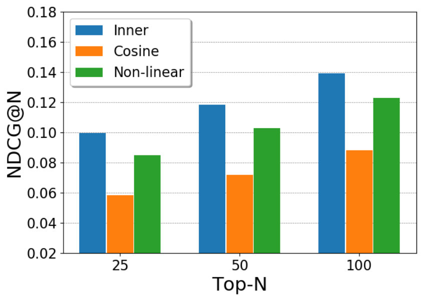
Effects of Dual Representation. We investigated the effects of dual representation in DualNPR. For three datasets, we consistently used inner product as the similarity function. As shown in Table 3, item-side triplets are useful to improve the accuracy of DualNPR: 7.6–50.8% in HR, 12.6–60.1% in NDCG, and 0.8–7.4% in AUC. Analyzing the results across multiple datasets, we found that the improvement owing to dual representation varies on the characteristics of the datasets. To demonstrate the significance of improvement, we computed the gain, which is the absolute difference of accuracy divided by the baseline accuracy. On average, the gain of MLLatest was 41.1% in HR, 52.9% in NDCG, and 3.1% in AUC. Likewise, the average gain of AMusic was 39.3% in HR, 49.6% in NDCG, and 7.4% in AUC. Meanwhile, the average gain of ML1M was 8.6% in HR, 14.0% in NDCG, and 0.8% in AUC. We observed that the performance gain in AMusic and MLLatest were much higher than those of ML1M. We believe that such a gap in improvement is strongly correlated with the sparsity of the datasets; AMusic (99.83%) and MLLatest (98.86%) are much more sparse than ML1M (95.53%). Therefore, we confirmed that our dual representation is beneficial to address the data sparsity problem.



(a) AMusic

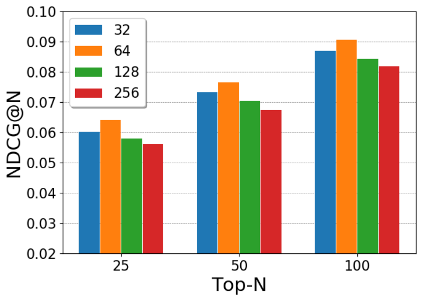


(b) MLLatest

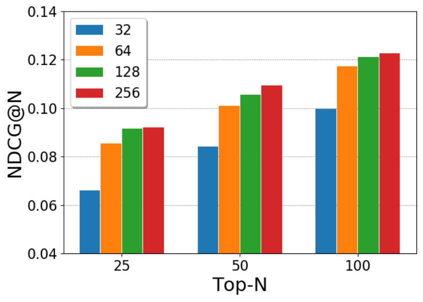


(c) ML1M

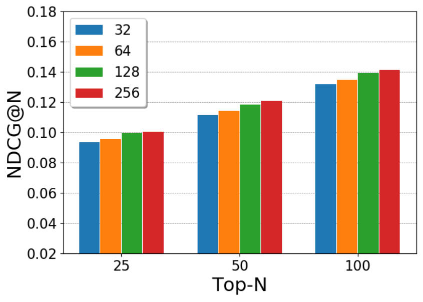
Figure 5: Comparisons of varying scoring functions in DualNPR.



(a) AMusic

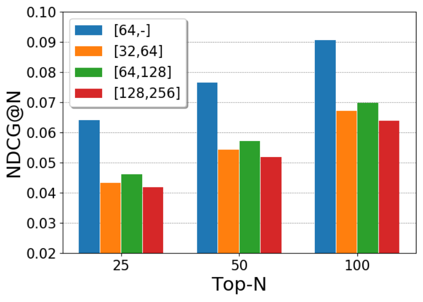


(b) MLLatest

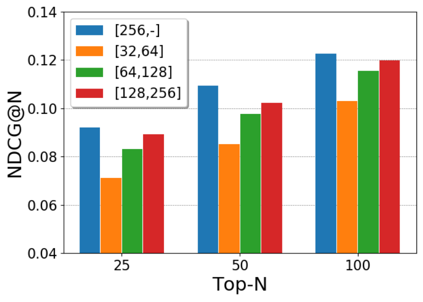


(c) ML1M

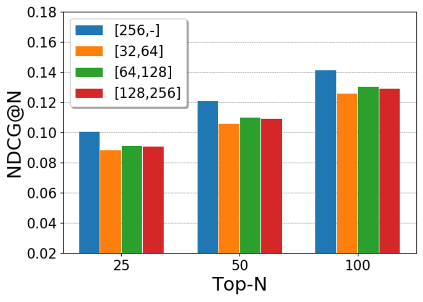
Figure 6: Comparisons of varying the size of embedding dimensions in DualNPR.



(a) AMusic



(b) MLLatest



(c) ML1M

Figure 7: Comparisons of varying the depth of hidden layers in DualNPR.

Effects of Negative Sampling. Our dynamic negative sampling method using an exponential function was compared with two baselines: random sampling and dynamic negative sampling using a linear function. As shown in Table 4, our sampling strategy is most effective at improving the accuracy of DualNPR for all three different datasets: 1.0–5.8% in HR, -2.9–8.6% in NDCG, and 0.3–0.8% in AUC. This means that our dynamic sampling with an exponential function tends to increase the gradient gap between positive and negative items, by selectively choosing negative items with high rankings from unobserved feedback. Compared to random sampling, our sampling method shows significant improvement gains by 7.3–50.6% in HR, 15.7–55.1% in NDCG, and 1.7–3.4% in AUC. Investigating these results over different datasets, we observe that our sampling strategy is more useful in handling the sparser dataset, such as AMusic. This justifies our choice of using the exponential function as it handles the sparse dataset more adequately, and improves the accuracy of top-*N* recommendation.

Effects of DMF. To determine the main components of DMF, we investigated various scoring functions, the size of embedding dimensions for users/items, and the depth of hidden layers. We observed several interesting results that were opposed to our expectations.

First, Figure 5 depicts the accuracy over three scoring functions in three datasets. It was observed that both inner product (MLLatest and ML1M) and cosine similarity (AMusic) are better than a non-linear inner product. We find that the choice of optimal scoring function varies on the dataset; it significantly influenced the accuracy. Although the non-linear scoring function is less useful in the three datasets, we still believe that the non-linear scoring function can be useful for other large-scale datasets.

Next, Figure 6 illustrates the accuracy at four different sizes of embedding dimension over the three datasets. The higher the dimension, the lower the accuracy. When the dimension was 64 in AMusic and 256 in both MLLatest and ML1M, we achieved the highest accuracy. The optimal size of the embedding dimensions tends to be closely related to the number of model parameters and the amount of positive feedback for each dataset. Note that the size of input representation multiplied by the size of embedding dimension decides the number of parameters in our model. Hence, when the size of input vector for three datasets was 16k, 9.7k, and 9.7k, respectively, the number of model parameters for three datasets and ML1M is 1M, 2.4M, and 2.4M, respectively. The optimal model parameters for AMusic are much less than MLLatest and ML1M because the amount of positive feedback is much less than other datasets. In other words, this is because, given the limited amount of positive feedback, increasing the model parameters can incur an overfitting problem.

Lastly, Figure 7 depicts the accuracy at various depths of hidden layers over the three datasets. The deeper the layer's depth, the lower the accuracy of DualNPR. From these observations, we conjecture that the correlations among users/items may not be highly complex and non-linear. In future work, we will investigate the effects of DMF over various large-scale datasets.

5 RELATED WORK

Much of the attention on personalized recommendation systems [24, 27] has shifted toward collaborative filtering (CF) utilizing implicit user feedback for top-N recommendation. Because an implicit dataset provides extremely sparse one-class feedback, it is inherently challenging to infer hidden user preferences from implicit feedback. This is also known as the one-class collaborative filtering (OCCF) problem [19]. To address this problem, existing work can be categorized into two groups: *point-wise* and *pair-wise* approaches.

5.1 Point-wise Approach

The point-wise approach aims at minimizing the absolute differences between actual values and predicted values. One of the popular models is weighted matrix factorization [10, 19]. Recently, with the success of DNNs [15], several studies [12, 35] have attempted to combine matrix factorization with DNNs such as autoencoders (AE) [16, 26, 28, 33], multi-layer perceptions (MLP) [1, 2, 8, 34, 36], convolutional neural networks (CNN) [7, 13], and recurrent neural networks (RNN) [9, 32]. Specifically, Wang et al. [28] combined a latent factor model with the autoencoders and Kim et al. [13] proposed a hybrid CF model by extracting textual features using CNN. Besides, Cheng et al. [1] and Covington et al. [2] proposed a wide-and-deep model where user features, item features, and additional contents are concatenated as input vectors. For sequential recommendation, Hidasi et al. [9] and Wu et al. [32] proposed RNN-based recommendation models. Although they outperform existing simple matrix factorization models, they have not optimized pairwise ranking for top-N recommendation.

5.2 Pair-wise Approach

The pair-wise approach aims to minimize a pairwise ranking loss in the ranked list. This approach is also used in other research domains such as computer vision [29, 30], information retrieval [11] and, natural language processing [3, 17]. Among recommendation models, the most representative model is BPR [23]. The objective function of BPR is to minimize a pairwise ranking loss of user-side triplets elicited from implicit user feedback. We categorize the variants of BPR into the following two groups.

BPR from Multiple Implicit Feedback. Pan et al. [20] extended BPR to accommodate different implicit feedback by adapting the confidence level of pairwise preferences. Later on, Qiu et al. [21] investigated the correlation between auxiliary feedback and target feedback and proposed an integrated model by considering the characteristics of different implicit feedback. Although this is valid for improving the quality of original BPR, they did not necessarily require heterogeneous implicit feedback. In contrast, our model naturally augments user- and item-side pairwise rankings from a user-item interaction matrix.

Integrating BPR with DNNs: As a hybrid approach, He et al. [6] proposed a visual BPR, where CNN is used for extracting visual features. Ding et al. [4] combined BPR with DNNs by extracting user features from social networks using CNN. Most recently, Niu et al. [18] proposed NPR by combining BPR with DNNs, and extended NPR to incorporate contextual information. These existing studies in this category [4, 6, 18] are most relevant to ours because they also utilize DNNs to improve the modeling power of BPR. However, we believe that our model can be viewed as a more generalized BPR model in the sense that we handle both user- and item-side pairwise rankings without any additional information. In particular, our model can be easily extended to other BPR variants.

6 CONCLUSION

In this paper, we have introduced a new CF model that uses implicit user feedback for top-N recommendation. To achieve this goal, we have proposed DualNPR that fully exploits both user- and item-side pairwise rankings using DNNs. The proposed model addresses the following key challenges faced in the existing work. (1) DualNPR overcomes the data sparsity problem by leveraging both user- and item-side triplets collected from a user-item interaction matrix. (2) DualNPR leverages DMF to learn the variability of latent user/item representations. (3) DualNPR effectively chooses negative feedback from unknowns by introducing dynamic negative sampling with an exponential function.

ACKNOWLEDGMENT

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (NRF-2018R1A2B6009135, NRF-2018R1A5A1081213, and NRF-2018R1A5A 1060031).

REFERENCES

[1] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, Rohan Anil, Zakaria Haque, Lichan Hong, Vihan Jain, Xiaobing Liu, and Hemal Shah. 2016. Wide & Deep Learning for Recommender Systems. In *International Workshop on Deep Learning for Recommender Systems DLRS@RecSys*. 7–10.

[2] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep Neural Networks for YouTube Recommendations. In *ACM International Conference on Recommender Systems (RecSys)*. 191–198.

[3] Arpita Das, Harish Yenala, Manoj Kumar Chinnakotla, and Manish Shrivastava. 2016. Together we stand: Siamese Networks for Similar Question Retrieval. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

[4] Daizong Ding, Mi Zhang, Shao-Yuan Li, Jie Tang, Xiaotie Chen, and Zhi-Hua Zhou. 2017. BayDNN: Friend Recommendation with Bayesian Personalized Ranking Deep Neural Network. In *ACM International Conference on Information and Knowledge Management (CIKM)*. 1479–1488.

[5] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *International Conference on Artificial Intelligence and Statistics, AISTATS*. 249–256.

[6] Ruining He and Julian McAuley. 2016. VBPR: Visual Bayesian Personalized Ranking from Implicit Feedback. In *International Conference on Artificial Intelligence (AAAI)*. 144–150.

[7] Xiangnan He, Xiaoyu Du, Xiang Wang, Feng Tian, Jinhui Tang, and Tat-Seng Chua. 2018. Outer Product-based Neural Collaborative Filtering. In *International Joint Conference on Artificial Intelligence (IJCAI)*. 2227–2233.

[8] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. In *International Conference on World Wide Web (WWW)*. 173–182.

[9] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2015. Session-based Recommendations with Recurrent Neural Networks. *CoRR* abs/1511.06939 (2015).

[10] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative Filtering for Implicit Feedback Datasets. In *IEEE International Conference on Data Mining (ICDM)*. 263–272.

[11] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry P. Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *ACM International Conference on Information and Knowledge Management (CIKM)*. 2333–2338.

[12] Alexandros Karatzoglou and Balázs Hidasi. 2017. Deep Learning for Recommender Systems. In *ACM International Conference on Recommender Systems (RecSys)*. 396–397.

[13] Dong Hyun Kim, Chanyoung Park, Jinoh Oh, Sungyoung Lee, and Hwanjo Yu. 2016. Convolutional Matrix Factorization for Document Context-Aware Recommendation. In *ACM International Conference on Recommender Systems (RecSys)*. 233–240.

[14] Yehuda Koren. 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. 426–434.

[15] Yann LeCun, Yoshua Bengio, and Geoffrey E. Hinton. 2015. Deep learning. *Nature* 521, 7553 (2015), 436–444.

[16] Sheng Li, Jaya Kawale, and Yun Fu. 2015. Deep Collaborative Filtering via Marginalized Denoising Auto-encoder. In *ACM International Conference on Information and Knowledge Management (CIKM)*. 811–820.

[17] Paul Neculoiu, Maarten Versteegh, and Mihai Rotaru. 2016. Learning Text Similarity with Siamese Recurrent Networks. In *International Workshop on Representation Learning for NLP (Rep4NLP@ACL)*. 148–157.

[18] Wei Niu, James Caverlee, and Haokai Lu. 2018. Neural Personalized Ranking for Image Recommendation. In *ACM International Conference on Web Search and Data Mining (WSDM)*. 423–431.

[19] Rong Pan, Yunhong Zhou, Bin Cao, Nathan Nan Liu, Rajan M. Lukose, Martin Scholz, and Qiang Yang. 2008. One-Class Collaborative Filtering. In *IEEE International Conference on Data Mining (ICDM)*. 502–511.

[20] Weike Pan, Hao Zhong, Congfu Xu, and Zhong Ming. 2015. Adaptive Bayesian personalized ranking for heterogeneous implicit feedbacks. *Knowl.-Based Syst.* 73 (2015), 173–180.

[21] Huihuai Qiu, Yun Liu, Guibing Guo, Zhu Sun, Jie Zhang, and Hai Thanh Nguyen. 2018. BPRH: Bayesian personalized ranking for heterogeneous implicit feedback. *Inf. Sci.* 453 (2018), 80–98.

[22] Steffen Rendle and Christoph Freudenthaler. 2014. Improving pairwise learning for item recommendation from implicit feedback. In *ACM International Conference on Web Search and Data Mining (WSDM)*. 273–282.

[23] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *International Conference on Uncertainty in Artificial Intelligence (UAI)*. 452–461.

[24] Francesco Ricci, Lior Rokach, and Bracha Shapira (Eds.). 2015. *Recommender Systems Handbook*. Springer.

[25] Ruslan Salakhutdinov and Andriy Mnih. 2007. Probabilistic Matrix Factorization. In *Annual Conference on Neural Information Processing Systems (NIPS)*. 1257–1264.

[26] Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie. 2015. AutoRec: Autoencoders Meet Collaborative Filtering. In *International Conference on World Wide Web Companion (WWW)*. 111–112.

[27] Koen Verstrepen, Kanishka Bhaduri, Boris Cule, and Bart Goethals. 2017. Collaborative Filtering for Binary, Positiveonly Data. *SIGKDD Explorations* 19, 1 (2017), 1–21.

[28] Hao Wang, Naiyan Wang, and Dit-Yan Yeung. 2015. Collaborative Deep Learning for Recommender Systems. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. 1235–1244.

[29] Liwei Wang, Yin Li, and Svetlana Lazebnik. 2016. Learning Deep Structure-Preserving Image-Text Embeddings. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 5005–5013.

[30] Liwei Wang, Yin Li, and Svetlana Lazebnik. 2017. Learning Two-Branch Neural Networks for Image-Text Matching Tasks. *CoRR* abs/1704.03470 (2017).

[31] Sheng Wang, Xiaobo Zhou, Ziqi Wang, and Ming Zhang. 2012. Please spread: recommending tweets for retweeting with implicit feedback. In *International Workshop on Data-driven User Behavioral Modelling and Mining from Social Media (DUBMMSM)*. 19–22.

[32] Chao-Yuan Wu, Amr Ahmed, Alex Beutel, Alexander J. Smola, and How Jing. 2017. Recurrent Recommender Networks. In *ACM International Conference on Web Search and Data Mining (WSDM)*. 495–503.

[33] Yao Wu, Christopher DuBois, Alice X. Zheng, and Martin Ester. 2016. Collaborative Denoising Auto-Encoders for Top-N Recommender Systems. In *ACM International Conference on Web Search and Data Mining (WSDM)*. 153–162.

[34] Hong-Jian Xue, Xinyu Dai, Jianbing Zhang, Shujian Huang, and Jiajun Chen. 2017. Deep Matrix Factorization Models for Recommender Systems. In *International Joint Conference on Artificial Intelligence (IJCAI)*. 3203–3209.

[35] Shuai Zhang, Lina Yao, and Aixin Sun. 2017. Deep Learning based Recommender System: A Survey and New Perspectives. *CoRR* abs/1707.07435 (2017).

[36] Shuai Zhang, Lina Yao, Aixin Sun, Sen Wang, Guodong Long, and Manqing Dong. 2018. NeuRec: On Nonlinear Transformation for Personalized Ranking. In *International Joint Conference on Artificial Intelligence (IJCAI)*. 3669–3675.

[37] Weinan Zhang, Tianqi Chen, Jun Wang, and Yong Yu. 2013. Optimizing top-n collaborative filtering via dynamic negative item sampling. In *International ACM SIGIR conference on research and development in Information Retrieval (SIGIR)*. 785–788.