Inf Retrieval J (2017) 20:109–131
DOI 10.1007/s10791-017-9295-9

Neural Semantic Personalized Ranking for item cold-start recommendation

Travis Ebesu[1] · Yi Fang[1]

Received: 2 September 2016 / Accepted: 13 February 2017 / Published online: 3 March 2017

Abstract Recommender systems help users deal with information overload and enjoy a personalized experience on the Web. One of the main challenges in these systems is the item cold-start problem which is very common in practice since modern online platforms have thousands of new items published every day. Furthermore, in many real-world scenarios, the item recommendation tasks are based on users' implicit preference feedback such as whether a user has interacted with an item. To address the above challenges, we propose a probabilistic modeling approach called Neural Semantic Personalized Ranking (NSPR) to unify the strengths of deep neural network and pairwise learning. Specifically, NSPR tightly couples a latent factor model with a deep neural network to learn a robust feature representation from both implicit feedback and item content, consequently allowing our model to generalize to unseen items. We demonstrate NSPR's versatility to integrate various pairwise probability functions and propose two variants based on the Logistic and Probit functions. We conduct a comprehensive set of experiments on two real-world public datasets and demonstrate that NSPR significantly outperforms the state-of-the-art baselines.

Keywords Recommender systems · Deep neural network · Implicit feedback · Pairwise learning

1 Introduction

Recommender systems have been established as a crucial tool for many Web applications to serve personalized recommendations to users. Successful systems span a wide variety of platforms, including Amazon's book recommendations, Netflix's movie recommendations,

✉ Yi Fang
yfang@scu.edu

Travis Ebesu
tebesu@scu.edu

[1] Department of Computer Engineering, Santa Clara University, Santa Clara, CA 95053, USA

and Pandora's music recommendations. A popular and effective approach to recommendations is collaborative filtering (CF), which focuses on finding users with similar interests and recommending items favored by the like-minded (Koren 2010). One of the fundamental problems arising when employing CF techniques is the item cold-start problem, which is caused by the system's incapability of dealing with new items due to the lack of relevant transaction history.

The problem of item cold-start is of great practical importance because modern online platforms publish thousands of new items everyday and effectively recommending them is essential for keeping the users continuously engaged. Content-based approaches, on the other hand, may still produce recommendations by using the descriptions of the items, but they tend to achieve lower accuracy. Combining CF and content becomes a common approach to item cold-start problems. Several hybrid latent factor models were proposed in the literature including collective matrix factorization (CMF) (Singh and Gordon 2008) and collaborative topic regression (CTR) (Wang and Blei 2011). The key idea is to obtain item latent factors from rating matrix and content matrix respectively and couple them in the shared latent space. These methods extend the traditional matrix factorization models by integrating content information, but the latent representation learned is often not effective especially when the content information is very sparse which is the case for many recommendation tasks where the item descriptions are usually quite short. The ineffectiveness may lie in the fact that these techniques can be viewed as shallow models in capturing latent topics from item descriptions and feedback information by applying simple transformations (often linear) on the observed data, while the ideal latent factors may have more complex relations with the observations.

Another challenge in many recommendation tasks is the presence of implicit feedback where users' explicit preferences (e.g., ratings) on items are unavailable. In the real world, often only implicit feedback is available to learn a recommendation model. Examples of implicit feedback are clicks, watched movies, played songs, purchases or assigned tags. Implicit feedback is tracked automatically and thus it is much easier to collect than explicit feedback. A characteristic of implicit feedback is that it is one-class, i.e. only positive observations are available. Moreover, the observed implicit feedback is generally very sparse, which makes the preference modeling even more challenging. As the result, existing solutions often deliver unsatisfactory recommendation accuracies.

On the other hand, deep learning models recently demonstrated great success for learning effective representations in various applications including computer vision, speech recognition, and natural language processing (Deng and Yu 2014; LeCun et al. 2015). However, the existing literature contains very few work on developing deep learning models for recommender systems, especially for addressing the cold-start problem with implicit feedback. In this paper, we propose a Neural Semantic Personalized Ranking (NSPR) probabilistic model by learning item representations using a deep neural network (DNN). To handle implicit feedback, we adopt pairwise probability functions that aim to discriminate between a small set of positive items and a very large set of all remaining items. In this way, items both with and without feedback will contribute to learning the ranking function and thus the data sparsity problem can be alleviated. DNN is used to map high-dimensional sparse text features into low-dimensional dense features in a latent semantic space. These low-dimensional features are tightly coupled with the latent factors learned from the pairwise probability, which allows two-way interactions between the content information and implicit feedback. The pairwise probability derived from the implicit feedback can guide the learning of feature representations. The learned features can further improve the predictive power of the pairwise model. The latent factors of new items can be inferred by applying the trained DNN to their content and then be used for item ranking. The contributions of the paper can be summarized as follows:

- We address the item cold-start recommendation task by inferring semantic representation of items using deep neural network and implicit user feedback. To the best of our knowledge, no prior work has exploited deep learning with implicit feedback for tackling cold-start problems.
- We propose a novel probabilistic generative modeling approach to characterize how a preference of items is observed based on implicit feedback. We derive two variants based on various pairwise probability functions including the Logistic and Probit functions.
- Extensive experiments on two public real-world datasets demonstrate that NSPR can significantly advance the state of the art.

2 Related work

Generally speaking, there are two main categories of recommendation tasks: rating prediction and item recommendation. The objective of rating prediction is to predict the rating that a user may give to an item that she has not interacted with before. Movie rating prediction in Netflix (Bennett and Lanning 2007) is such an example. For item recommendation, recommender systems provide a user with a ranked list of items that she might prefer. Examples include product recommendation in Amazon (Linden et al. 2003) and point-of-interest recommendation in location-based social networks (Cheng et al. 2012). This paper focuses on the item cold-start recommendation task. In the following subsections, we briefly review three categories of the existing work relevant to ours.

2.1 Deep learning in collaborative filtering

There exists few work on deep learning for recommender systems. Salakhutdinov et al. (2007) proposed restricted Boltzmann machines (RBM) to model the ratings of movies. Georgiev and Nakov (2013) extended this work in a unified non-IID framework. Recently, auto-encoders have been a popular architecture to address recommender systems. Auto-encoders are a feedforward neural network which constructs a compressed representation by forming a bottleneck in the architecture before attempting to recover the models initial inputs. AutoRec (Sedhain et al. 2015) demonstrated the effectiveness of a simple auto-encoder to predict movie ratings directly, improving upon previous deep learning models previously applied to recommender systems. Incorporating corrupt inputs or noise to auto-encoders further improved performance and as a result, many variants utilizing auto-encoders have since emerged. One such example is Collaborative denoising autoencoders (CDAE) (Wu et al. 2016) which integrates user latent factors with an auto-encoder to address Top-N Recommender Systems examining both pointwise and pairwise loss functions. Strub and Jeremie (2015) establish a methodology capable of training deep architecture of stacked denoising auto-encoders by interpolating the corrupt input and reconstruction error as a loss function. Neural network matrix factorization (NNMF) (Dziugaite and Roy 2015) take a different approach by replacing the traditional inner product of matrix factorization with a function learned from a feedforward neural network. CoFactor (Liang et al. 2016) jointly factorizes the ratings matrix and the shifted positive pointwise mutual information (SSPMI) item embeddings matrix. Factoring the SSPMI matrix has been shown to be equivalent to word2vec for item co-occurrence embeddings (Levy and Goldberg 2014). These methods do not incorporate content information and thus cannot address the cold-start problem.

Some recent work by Van den Oord et al. (2013) and Wang and Wang (2014) tackled music recommendation by a two-step approach: matrix factorization is used to obtain the latent factors for items and then conventional convolutional neural network (CNN) is applied to learn feature representation for content information by treating these latent factors as the output. In other words, the deep learning components of their models are deterministic and only loosely coupled with the matrix factorization of the rating matrix. They do not exploit the interactions between content information and ratings.

Cheng et al. (2016) tackle mobile app recommendation by jointly training a generalized linear model and DNN on hand engineered user demographic and implicit app installations. The DNN produces diverse mobile app recommendations while the linear model mediates overgeneralization to irrelevant recommendations. The joint recommendation provides a middle ground between the two. Similarly, Neural Collaborative Filtering (NCF) (He et al. 2017) partners the output of a multi-layer perceptron (MLP) concatenated with the latent factors from matrix factorization (MF) applying a nonlinear transformation to produce a local interaction before performing the final recommendation. The MLP and MF each retain separate embedding spaces for the user and item latent factors accommodating the required complexity for the task at hand. NCF may appear similar to our proposed model; however, NCF lacks the ability to handle content information hence cannot be used in the cold-start problem. Recurrent Recommender Networks (RRN) (Wu et al. 2017) represent user latent and item latent factors with two recurrent neural networks (RNN) to capture the temporal aspect of movie recommendation. Collectively, the RNNs hidden states represent the user's preference and ratings at each time interval while additional stationary factors are maintained to handle a user's long-standing preferences. Jing and Smola (2017) endow an RNN with survival analysis to predict the future return of a given user. The RNN addresses the temporal aspect consulting previous hidden states with the survival rate to address the user's return time. Zheng et al. (2017) portray user behavior and item properties with parallel CNNs on user reviews and item reviews respectively, before employing a final shared interaction layer. Zhang et al. (2016a) leverage textual, structural and visual knowledge bases with convolutional and denoising auto-encoders to enhance the latent factor model. Zhang et al. (2016b) tackle click-through-rates by modifying the input layer of a feedforward neural network to perform different types of transformations over multi-field categorical data. Three transformations are proposed based on factorization machines, restricted Boltzmann machines (RBM) and denoising auto-encoders. However, these methods lack adaptability to our problem or impose additional feature engineering such as knowledge bases or user demographics.

Collaborative deep learning (CDL) (Wang et al. 2015) a hierarchical Bayesian model, is proposed to tightly couple deep representation learning for the content information and collaborative filtering for the rating matrix, allowing two-way interaction between the two. Collaborative deep ranking (CDR) (Ying et al. 2016) later extends CDL to include a pairwise loss. Deep collaborative filtering (DCF) (Li et al. 2015), mitigates the computational overhead in deep learning by marginalizing a denoising auto-encoder in order to obtain a closed form solution. The architecture consists of an item and user auto-encoder for content information coupled with a latent factor model. CDL and DCF models both share some similarities with NSPR. However, they directly predict user ratings and lack the ability to address implicit feedback which is pervasive in modern recommender systems.

While CDR does use a pairwise loss for implicit feedback it does not exploit a latent factor model.

NSPR has notable differences from the existing work. First of all, no prior work has studied the cold-start problems by coupling deep semantic representation with user feedback. Secondly, many previous deep models in recommender systems use denoising auto-encoders to learn a feature representation from content, while NSPR utilizes a deep neural network which allows to model the latent semantic space directly without modeling the recovery of input as the auto-encoders do. We demonstrate the effectiveness of using a DNN to learn robust feature representations without complex preprocessing data transformations. Last but not the least, NSPR utilizes stochastic gradient descent for parameter estimation, which is often more scalable for large datasets than the batch estimation methods (Bottou 2010) which were used in the existing deep learning based recommendation models such as CDL and DCF.

2.2 Cold-start problem

Cold-start problems are prevalent in recommender systems. They are often alleviated by utilizing content information. Word-based similarity methods (Pazzani and Billsus 2007) recommend items based on textual content similarity in word vector space. Collaborative topic regression (CTR) couples a matrix factorization model with probabilistic topic modeling to generalize to unseen items (Wang and Blei 2011). Collective matrix factorization (CMF) (Singh and Gordon 2008) simultaneously factorizes both rating matrix and content matrix with shared item latent factors. SVDFeature (Chen et al. 2012) combines content features with collaborative filtering. The latent factors are integrated with user, item, and global features. SVDFeature demonstrated the state-of-the-art performance in benchmark evaluations.

2.3 Implicit feedback

Matrix factorization has been adapted to learn from implicit feedback for recommendation. Regularized least-square optimization with case weights is proposed in Hu et al. (2008) and Pan et al. (2008). One of the most effective techniques is based on Bayesian personalized ranking (BPR) (Rendle et al. 2009) which has been shown to provide strong results in many item recommendation tasks. Several extensions of BPR include pairwise interaction tensor factorization (Rendle and Schmidt-Thieme 2010), multi-relational matrix factorization (Krohn-Grimberghe et al. 2012), and non-uniformly sampled items (Gantner et al. 2012). Pan and Chen (2013) proposed group Bayesian personalized ranking (GBPR) via introducing group preference. Rendle and Freudenthaler (2014) incorporate an adaptive sampling method to speed up learning convergence rate by utilizing the fact that the implicit feedback matrix follows a tailed distribution of item popularity. All of the above BPR based models do not consider any content information and cannot address the cold-start problems.

3 Neural Semantic Personalized Ranking

We take a pairwise approach to item recommendation by assuming that a user prefers the items that she has interacted with rather than those items that she has not interacted with. This assumption is more reasonable than the pointwise assumption which treats all observed entries in the user-item interaction/feedback matrix as positive examples and all missing entries as negative examples.

Formally, given user $i \in U$, we use $j^+ \in I_i^+$ to denote a positive item (i.e., interacted/observed item) where I_i^+ is the set of all positive items for user i. Similarly, we use $j^- \in V \backslash I_i^+$ for a negative item (i.e., uninteracted/unobserved item) where V is the set of all items. Since item j^+ is preferred over item j^-, we can form a preference instance $(i, j^+, j^-) \in D_S$ where $D_S = \{(i, j^+, j^-) | i \in U, j^+ \in I_i^+, j^- \in V \backslash I_i^+\}$ is the whole set of preference instances. The total number of preference triplets is quadratic in the number of items. Thus, we sample from D_S for training instead of going over the complete set of item pairs (Sect. 4.2 gives the details about our sampling strategy). Table 1 lists the main notations used in the paper.

3.1 Probabilistic generative modeling

We propose Neural Semantic Personalized Ranking (NSPR) by tightly incorporating a deep neural network (DNN) (Hinton et al. 2012) to learn effective feature representation from item content. The DNN architecture maps the raw text features into the features in a semantic space. The input (raw text features) to the DNN is a high dimensional term vector, e.g., TF-IDF of terms in the item content, and the output of the DNN is a concept vector in a low-dimensional semantic feature space. Formally, we denote $\mathbf{d}_j$ as the input term vector, $\mathbf{y}_j$ as the output vector, l as the lth hidden layer ($l \in [1, L-1]$). $\mathbf{a_l}$, $\mathbf{W_l}$ and $\mathbf{b_l}$ are the activation output, weight matrix and bias vector respectively. We have

Table 1 Notations

i, j	Index for user and item respectively
U, V	User set and item set respectively
$\mathbf{u}_i$	Latent factor for user i
$\mathbf{v}_j$	Latent factor for item j
$r(i, j)$	Ranking score of item j for user i
I_i^+	Set of all positive items for user i
(i,j^+,j^-)	A preference triplet indicating user i prefers item j^+ over item j^-
D_s	Set of preference triplet instances
D_i	Set of preference instances for user i
$\mathbf{d}_j$	Input content vector for item j
$\mathbf{y}_j$	Output latent feature vector for item j
$\mathbf{a}_l$, $\mathbf{W}_l$, $\mathbf{b}_l$	Activation output, weight matrix and bias vector at the lth layer in DNN respectively
σ_u^2	Variance in user prior distribution
σ_v^2	Variance of noise in latent item factor
σ_r^2	Variance of noise in ranking score
K	Number of latent factors
L	Number of layers in DNN
N	Size of vocabulary

$$\begin{aligned} \mathbf{a}_{1,j} &= \mathbf{W}_1 \mathbf{d}_j \\ \mathbf{a}_{l,j} &= \psi(\mathbf{W}_l \mathbf{a}_{l-1,j} + \mathbf{b}_l) \\ \mathbf{y}_j &= \mathbf{W}_L \mathbf{a}_{L-1,j} + \mathbf{b}_L \end{aligned}$$

where we use the tanh as the activation function ψ at the hidden layers and the identity function for the output layer.

$$\psi(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}} \tag{1}$$

The output concept vector $\mathbf{y}_j$ is used to calibrate the latent item factor vector $\mathbf{v}_j$ learned from the feedback matrix. On the other hand, the weights and bias in DNN are learned with the guidance of $\mathbf{v}_j$. In other words, $\mathbf{y}_j$ and $\mathbf{v}_j$ are tightly coupled, which allows two-way interactions between the content information and implicit feedback. Specifically, NSPR can be viewed as a probabilistic modeling approach with the generative process described as follows (the graphical model representation of NSPR is shown in Fig. 1).

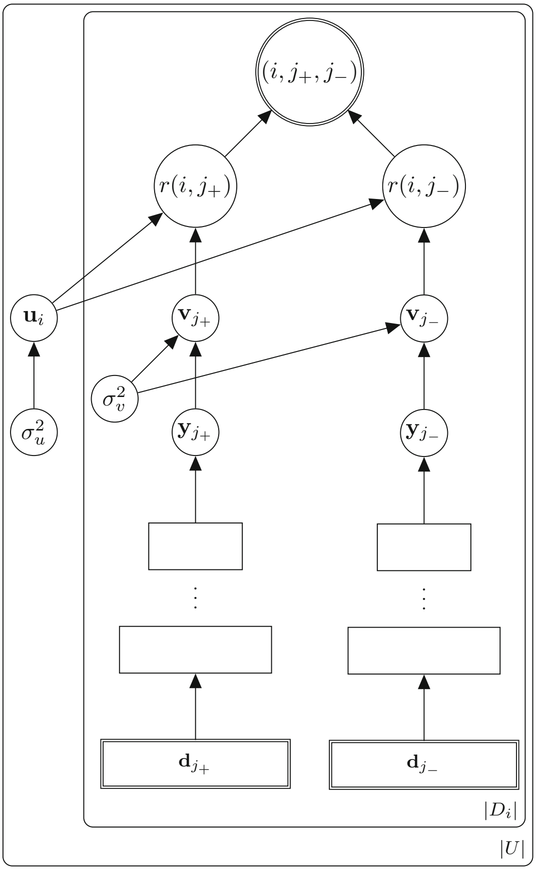

Fig. 1 Graphical model representation of NSPR. The *double circled nodes* represent observed variables and other nodes are latent variables

1. For each item j,
   (a) Map high-dimensional sparse text feature vector $\mathbf{d}_j$ into low-dimensional dense features $\mathbf{y}_j$ via DNN
   (b) Draw a latent item offset vector from normal distribution:
   $$\epsilon_j \sim \mathcal{N}(\mathbf{0}, \sigma_v^2 \mathbf{I}_K) \tag{2}$$
   (c) Set the latent item vector to be
   $$\mathbf{v}_j = \mathbf{y}_j + \epsilon_j$$
2. For each user i, draw a latent user factor
   $$\mathbf{u}_i \sim \mathcal{N}(\mathbf{0}, \sigma_u^2 \mathbf{I}_K)$$
3. For item j given user i, calculate the ranking score $r(i,j) = f(\mathbf{u}_i, \mathbf{v}_j)$. For user i, item j^+ and j^-, form the preference triplet (i,j^+,j^-) with the probability $\mathcal{S}(r(i,j^+) - r(i,j^-))$ where $\mathcal{S}$ is a sigmoid 'S' shape class of functions.

Here $\mathcal{S}\Big(r(i,j^+) - r(i,j^-)\Big)$ defines a pairwise probability that is a monotonically non-decreasing function with respect to the argument $r(i,j^+) - r(i,j^-)$. The intuitive explanation is that if item j^+ is preferred over j^- for user i, the difference between their ranking scores $r(i,j^+)$ and $r(i,j^-)$ is maximized given the monotonically non-decreasing function $\mathcal{S}(x)$. As a result, item i is more preferable than item j. In Sect. 3.3, we define two variants over the NSPR framework drawing on the Logistic and Probit probability functions.

In this paper, we set the ranking score as $r(i,j) = f(\mathbf{u}_i, \mathbf{v}_j) = \mathbf{u}_i^T \mathbf{v}$, which leads to

$$r(i,j^+) - r(i,j^-) = \mathbf{u}_i^T \left(\mathbf{v}_{j+} - \mathbf{v}_{j-}\right)$$

It is worth noting that the output of the DNN serves as a bridge between the feedback and content information, which is the key that enables NSPR to simultaneously learn an effective feature representation and capture the implicit preference relations between items. The low-dimensional output obtained by DNN is tightly coupled with the latent factors learned from the pairwise probability. The pairwise probability derived from the implicit feedback can guide the learning of feature representations. The learned features can further improve the predictive power of the pairwise ranking model. Thus, the low-dimensional feature representation obtained by DNN captures the latent semantic of item content while being predictive for item ranking, which is very desirable for addressing the item cold-start problem.

3.2 Parameter estimation

Based on the NSPR framework above, the posterior likelihood of observing all the preference triplets is:

$$L = \prod_i \prod_{j^+,j^-} \mathcal{S}\Big(r(i,j^+) - r(i,j^-)\Big) \prod_{j^+,j^-} \mathcal{N}\left(\mathbf{v}_j | \mathbf{y}_j, \sigma_v^2 \mathbf{I}\right) \prod_i \mathcal{N}\left(\mathbf{u}_i | \mathbf{0}, \sigma_u^2 \mathbf{I}\right)$$

By taking the log of the likelihood and simplifying we obtain

$$\mathcal{L} = \sum_i \sum_{j^+, j^-} \log \mathcal{S}\Big(r(i,j^+) - r(i,j^-)\Big) \\ - \frac{1}{2\sigma_v^2} \sum_{j^+, j^-} ||\mathbf{v}_j - \mathbf{y}_j||_2^2 - \frac{1}{2\sigma_u^2} \sum_i ||\mathbf{u}_i||_2^2 \tag{3}$$

The parameters to be learned include latent factors $\mathbf{u}_i$ and $\mathbf{v}_j$, and the weights $\mathbf{W}_l$ and bias $\mathbf{b}_l$ in the DNN. The second term in the objective function above is to encode a deep neural network using the latent item vectors $\mathbf{v}_j$ as the target.

We use Stochastic Gradient Descent (SGD) to obtain the Maximum A Posteriori (MAP) estimate. For a given triplet of latent factors $(\mathbf{u}_i, \mathbf{v}_{j^+}, \mathbf{v}_{j^-})$, we compute the stochastic gradients given the current outputs of the DNN (i.e. $\mathbf{y}_j$).

$$\frac{\partial \mathcal{L}}{\partial \mathbf{u}_i} = \frac{1}{\mathcal{S}} \frac{\partial \mathcal{S}}{\partial x} \left(\mathbf{v}_{j^+} - \mathbf{v}_{j^-} \right) - \frac{1}{\sigma_u^2} \mathbf{u}_i \tag{4}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{v}_{j+}} = \frac{1}{\mathcal{S}} \frac{\partial \mathcal{S}}{\partial x} \mathbf{u}_i - \frac{1}{\sigma_v^2} \left(\mathbf{v}_{j+} - \mathbf{y}_{j+} \right) \tag{5}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{v}_{j-}} = -\frac{1}{\mathcal{S}} \frac{\partial \mathcal{S}}{\partial x} \mathbf{u}_i - \frac{1}{\sigma_v^2} \left(\mathbf{v}_{j-} - \mathbf{y}_{j-} \right) \tag{6}$$

where $\frac{\partial \mathcal{S}}{\partial x}$ is the stochastic gradient of the pairwise probability $\mathcal{S}(\cdot)$ with respect to its input ranking score preference. Section 3.3 will derive $\frac{\partial \mathcal{S}}{\partial x}$ for various forms of $\mathcal{S}(\cdot)$.

Given the current $\mathbf{v}_{j^+}$ and $\mathbf{v}_{j^-}$, we can then update the weights $\mathbf{W}_l$ and biases $\mathbf{b}_l$ for each layer of the DNN using the backpropagation algorithm (Rumelhart et al. 1988). The stochastic gradients of the likelihood with respect to $\mathbf{W}_l$ and biases $\mathbf{b}_l$ are as follows:

$$\begin{aligned} &\frac{\partial \mathcal{L}}{\partial \mathbf{W}_l} = \delta_{l,j} \mathbf{a}_{l,j}^T \quad \text{and} \quad \frac{\partial \mathcal{L}}{\partial \mathbf{b}_l} = \delta_{l,j} \\ &\text{where} \quad \delta_{l,j} = \mathbf{W}_{l+1}^T \delta_{l+1,j} \odot (\mathbf{1} - \mathbf{a}_{l,j} \odot \mathbf{a}_{l,j}) \\ &\text{and} \quad \delta_{L,j} = \mathbf{v}_j - \mathbf{y}_j \end{aligned}$$

where $\odot$ is the element-wise product. The algorithm iterates over the gradient updates for each preference triplet (i, j^+, j^-) until convergence. Section 4.2 discusses the details about the setting of the algorithm.

3.3 Pairwise probability

NSPR seamlessly integrates with a multitude of pairwise probability functions for $\mathcal{S}(\cdot)$. In our case, the two pairwise functions we chose can also be interpreted as cumulative distribution functions. We define two variants over the NSPR framework to demonstrate its capabilities.

3.3.1 Logistic probability

One of the most widely used sigmoid functions is the Logistic function, defined as

$$\mathcal{S}(x) = \frac{1}{1 + \exp(-x)} \tag{7}$$

It is worth noting in this setting, if σ_v^2 goes to infinity, the maximization of the objective function Eq. (3) is degenerated to the BPR-MF model (Rendle et al. 2009). The use of non-zero σ_v^2 in NSPR enables the coupling between the semantic item representation learned by the deep neural network and the latent item factors learned from the pairwise implicit feedback. This tight coupling is missing in the BPR based models.

Computing the stochastic gradient, we obtain the following

$$\frac{\partial \mathcal{S}}{\partial x} = \Big(1 - \mathcal{S}\big(r(i,j^+) - r(i,j^-)\big)\Big)\mathcal{S}\big(r(i,j^+) - r(i,j^-)\big) \tag{8}$$

Plugging Eq. (8) into Eqs. (4), (5) and (6), we obtain the parameter estimation update for the Logistic variant of NSPR, called as NSPR-L.

3.3.2 *Probit probability*

In statistics, closely related to the Logistic function are the Probit function and Probit model (McCullagh and Nelder 1989). The Logistic and Probit are both sigmoid functions with a domain between 0 and 1, which makes them both quantile functions—i.e., inverses of the cumulative distribution function (CDF) of a probability distribution. In fact, the Logistic is the quantile function of the Logistic distribution, while the Probit is the quantile function of the Gaussian distribution. We derive the Probit variant of NSPR, denoted as NSPR-P, by setting $\mathcal{S}(x) = \Phi(x)$ as the cumulative distribution function of the Gaussian distribution as follows:

$$\Phi(x) = \int_{-\infty}^{x} \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) dx$$

We can then obtain the stochastic gradient of the objective function as follows:

$$\frac{\partial \mathcal{S}}{\partial x} = \mathcal{N}\big(r(i,j^+) - r(i,j^-)\big)$$

where

$$\mathcal{N} = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right).$$

For simplicity we set $\mu = 0$ and $\sigma^2 = 1$ yielding the standard normal Gaussian distribution. Figure 2 plots the Logistic, Probit, and Heaviside step functions. As we can see, these functions have a similar ‘S’ shape. The Logistic has a slightly flatter tail while the Probit curve approaches the axes more quickly. In the Probit function, as we increase the variance the curve will become flatter and elongated. The experiments in Sect. 4 compare the performance of the two variants of NSPR.

3.4 Prediction for cold-start items

Once the NSPR model is trained, the parameters are used to calculate the ranking score $r(i, j)$ for item j given user i. The items are ranked in descending order of $r(i, j)$ for

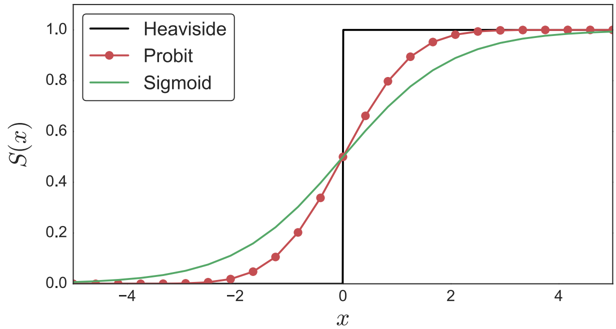

Fig. 2 Logistic and Probit pairwise probability functions in NSPR

providing personalized recommendation. Similar to Wang and Blei (2011), we use the MAP point estimates of parameters to calculate the predicted ranking score

$$r(i,j) \approx (\mathbf{u}_i^*)^T(\mathbf{y}_j + \epsilon_j^*) = (\mathbf{u}_i^*)^T(\mathbf{v}_j^*) \tag{9}$$

where $\mathbf{u}_i^*$ and $\mathbf{v}_j^*$ are the point estimates by SGD in Sect. 3.2 for the random variables $\mathbf{u}$ and $\mathbf{v}$. $\mathbf{y}$ is deterministic mapped from the content feature vector $\mathbf{d}$.

For the cold-start problem when the item j is unseen in the training data, we set the noise offset ϵ_j^* in Eq. (9) to be zero and obtain the predicted ranking score as follows

$$r(i,j) \approx (\mathbf{u}_i^*)^T\left(\mathbf{W}_L\mathbf{a}_{L-1,j} + \mathbf{b}_L\right) \tag{10}$$

where $\mathbf{W}_L\mathbf{a}_{L-1,j} + \mathbf{b}_L$ is the output of DNN based on item content input $\mathbf{d}_j$.

4 Experiments

4.1 Datasets

We evaluate our model on two public datasets from CiteULike[1] and Yahoo! Movies.[2] CiteULike is a web service that allows users to save and share citations to academic papers. The first dataset *citeulike-a*[3] (Wang and Blei 2011) contains 5551 users, 16,980 items with 204,987 positive entries. Implicit feedback is encoded as positive if the user has the item in their personal library and encoded as negative otherwise. The second dataset, *Yahoo! Movies* consists of users rating movies on a scale of 1–5 with a short synopsis. To be consistent with the implicit feedback setting, we extract only positive ratings (rating 5) for training and testing. After removing movies without a synopsis, this yields 7642 users, 11,915 items, and 221,367 positive ratings. The characteristics of the dataset are summarized in Table 2. It is worth noting that *citeulike-a* is sparser in ratings and has over twice the number of average words per a document while the contrary is true for *Yahoo! Movies*. Similar to Wang and Blei (2011) and Wang et al. (2015), we preprocess the data

[1] http://www.citeulike.org.

[2] R4 - Yahoo! Movies User Ratings and Descriptive Content Information, v.1.0 http://webscope.sandbox.yahoo.com/.

[3] http://www.cs.cmu.edu/∼chongw/data/citeulike.

Table 2 Dataset statististics

	citeulike-a	*Yahoo! Movies*
Users	5551	7642
Items	16,980	11,915
Ratings	204,987	22,136
Sparsity (%)	99.78	99.76
Vocabulary size	68,911	39,664
Avg. words/document	187.97	68.26
Avg. ratings/user	37.92	118.50

by removing the users with fewer than 3 positive entries, concatenating the title and abstract (movie synopsis), removing stopwords, stemming and construct our vocabulary from the top N terms based on TF-IDF then use raw term counts. We randomly hold out 20% of the items for testing and the remaining 80% of the items are used for training. The split of data yields the cold-start setting since the items in each set are disjoint from the other sets, and are new items for the users. We set the vocabulary size (N) to 8000 and 20,000 for the *citeulike-a* and *Yahoo! Movies* datasets, respectively.

4.2 Baselines and settings

We use the following baselines for comparison in the experiments. They are the state-of-the-art recommendation algorithms for recommendation tasks and consider content information a requirement for an algorithm to address the item cold-start problem.

- SVDFeature (Chen et al. 2012), which performs feature-based matrix factorization allowing for additional content and relationships.
- Collective matrix factorization (CMF) (Singh and Gordon 2008), which simultaneously factors multiple matrices to learn integrating relations between them.
- Collaborative topic regression (CTR) (Wang and Blei 2011), which combines probabilistic topic modeling with a latent factor model.
- Collaborative deep learning (CDL) (Wang et al. 2015), which creates a deep feature representation using stacked denoising auto-encoders with CTR.
- Neural Semantic Personalized Ranking (NSPR) with two variants: Logistic (NSPR-L) and Probit (NSPR-P) which we proposed in Sect. 3.

We select all hyperparameters by cross-validation grid search, holding out 10% of the training data to create a separate validation set. We then tune hyperparameters according to Recall@300 achieved on the validation set. In our experimental results, we utilize both the training and validation sets as training data. For SVDFeature, we use the ranking setting and found good results when λ_u and λ_v are set to 0.04. In CMF, we set both matrices (rating and item content) to the sparse setting and 0.1 and 0.05 for the ratings and item content matrices respectively. CTR performed best when we set $a = 1$, $b = 0.01$, $\lambda_u = 0.1$, and $\lambda_v = 10$. CDL performed best with the architecture "200–200–K–200–200" with $\lambda_v = 10$, $\lambda_u = 1$ and $\lambda_n = 100$.

For the SGD algorithm of our NSPR models, we use the adaptive subgradient method (AdaGrad) (Duchi et al. 2011) to schedule the learning rate with the initial value of 0.1. The regularization parameter σ_u^2 of latent user factors are set to be 9. We randomize the preference triplets (i, j^+, j^-) for SGD training by uniformly randomly sampling a user from

U, a positive item from I_i^+, and a negative item from $V \backslash I_i^+$, respectively. This sampling strategy reduces the chance of updating the same user-item combination in consecutive iterations, which otherwise may lead to poor convergence (Rendle et al. 2009). The initial values of the parameters in the SGD algorithm are uniformly randomly sampled from [0, 1] and the stopping criteria is when the relative change of the likelihood function is less than 0.01%. The default number of nodes for each hidden layer is 256. We set the default parameters for both variants on the *citeulike-a* dataset with 128 latent factors, σ_v^2 to 500 and dropout to 0.1 with two hidden layers. In the *Yahoo! Movies* dataset, both variants use two hidden layers with $K = 16$. We set σ_v^2 to 9 and 200 for NSPR-L and NSPR-P respectively. We use the default parameter values in the experiments unless otherwise specified.

4.3 Evaluation metrics

The accuracy of a recommendation model is measured by using three commonly used metrics, namely Mean average precision (MAP), Normalized discounted cumulative gain (NDCG), and Recall (R) (Manning et al. 2008). MAP is widely adopted for evaluation of item recommendation. Because users are usually interested in a few top-ranked items, NDCG@N is used to compare the top-N recommendation performance. We also use Recall because the feedback information is implicit. Precision oriented metrics such as MAP and NDCG may not be sufficient since a negative entry could be caused by the fact that the user is not interested in the item, or that the user is not aware of its existence.

4.4 Results

4.4.1 Number of latent factors

Selecting the optimal number of latent factors and hidden layers can have a devastating effect on performance as we demonstrate in this section. Varying these hyperparameters may introduce noise causing difficulty in isolating the actual effect. To account for the variance, we perform tenfold cross-validation by splitting the items into ten equal parts. We use ninefolds as training data and the final fold as testing such that we yield a cold-start setting as described earlier in Sect. 4.2. We repeat this process ten times each with a different test fold and report the average Recall@300 and NDCG@10.

Figure 3 illustrates the effect of varying the number of latent factors ($K = 16, 32, 64, 128$, and 256) and hidden layers ($L = 1, 2, 3, 4$) for both NSPR variants reporting the mean Recall@300 and NDCG@10 for the *citeulike-a* dataset. In both variants, as the number of latent factors increases a corresponding climb in performance is seen on both metrics despite the number of hidden layers. Each particular configuration obtains peak performance on both metrics at 128 latent factors with the exception of NSPR-L where the curve continues to increase with 256 latent factors and four hidden layers. With respect to the number of hidden layers in the DNN, a single hidden layer struggles to capture the intricate non-linear semantics. The optimal Recall and NDCG occurs at two and three hidden layers where sufficient modeling capacity exists. NSPR-L (bottom) with two hidden layers obtains the best performance for Recall@300 with 128 latent factors. NSPR-P (top) simultaneously performs the best on Recall and NDCG with three hidden layers and 128 latent factors. Multiple parameter configurations demonstrate competitive performance across both metrics. We report the variance over the tenfolds in Table 3 where we find the variance is relatively small. Overall, lower fluctuations were reported with

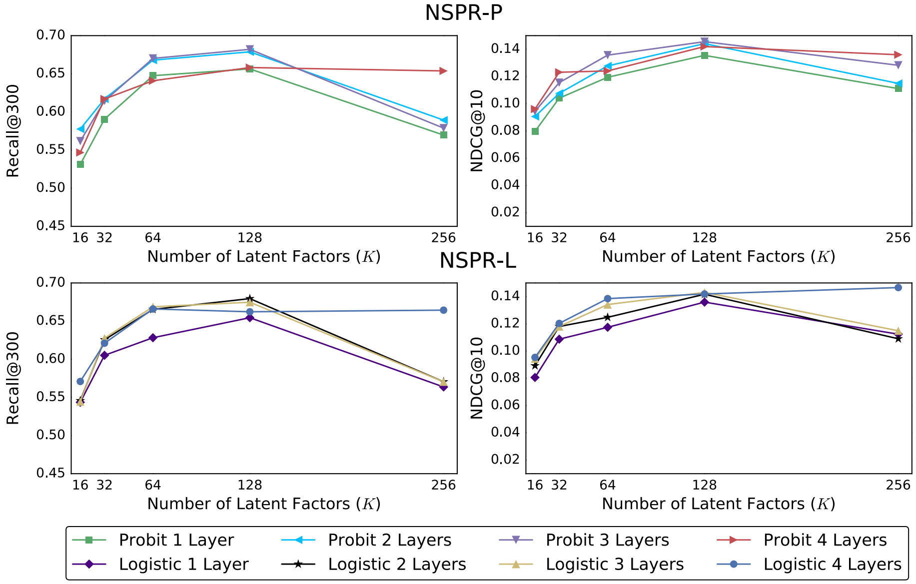

Fig. 3 Recall@300 (*left*) and NDCG@10 (*right*) for varying number of latent factors (K) and hidden layers (L) averaged over tenfolds on the *citeulike-a* dataset

Table 3 Variance over tenfolds of NSPR variants for varying the number of latent factors (K) and hidden layers (L) on the *citeulike-a* dataset

	Recall@300					NDCG@10				
	16	32	64	128	256	16	32	64	128	256
P/1	0.0020	0.0022	0.0003	0.0012	0.0152	0.0001	0.0002	0.0001	0.0001	0.0011
P/2	0.0009	0.0010	0.0002	0.0003	0.0072	0.0001	0.0002	0.0002	0.0000	0.0008
P/3	0.0026	0.0017	0.0003	0.0003	0.0242	0.0001	0.0002	0.0003	0.0001	0.0012
P/4	0.0046	0.0020	0.0068	0.0025	0.0020	0.0002	0.0002	0.0008	0.0005	0.0004
L/1	0.0021	0.0007	0.0032	0.0011	0.0040	0.0001	0.0000	0.0002	0.0000	0.0001
L/2	0.0040	0.0003	0.0003	0.0001	0.0039	0.0002	0.0001	0.0002	0.0001	0.0006
L/3	0.0096	0.0007	0.0003	0.0003	0.0104	0.0003	0.0002	0.0002	0.0002	0.0010
L/4	0.0030	0.0011	0.0003	0.0008	0.0002	0.0004	0.0002	0.0002	0.0003	0.0001

We denote P/1 to indicate NSPR-Probit with one hidden layer and similarly, L/2 to indicate NSPR-Logistic with two hidden layers

latent factors in the range of [32, 64] and two hidden layers in both variants. Conversely, the highest volatility is achieved with three hidden layers and 256 latent factors

In Fig. 4, both NSPR variants show the best performance with two hidden layers and latent factors in the range of [32, 64] on the *Yahoo! Movies* dataset. In general, NSPR with a single hidden layer lacks the capability to model the complex relations present. As three or more hidden layers are used a diminish in performance may suggest overfitting. NSPR-P with one and two hidden layers show similar performance and a balance between Recall and NDCG. Illustrating an equilibrium between the pairwise, latent factors and DNN

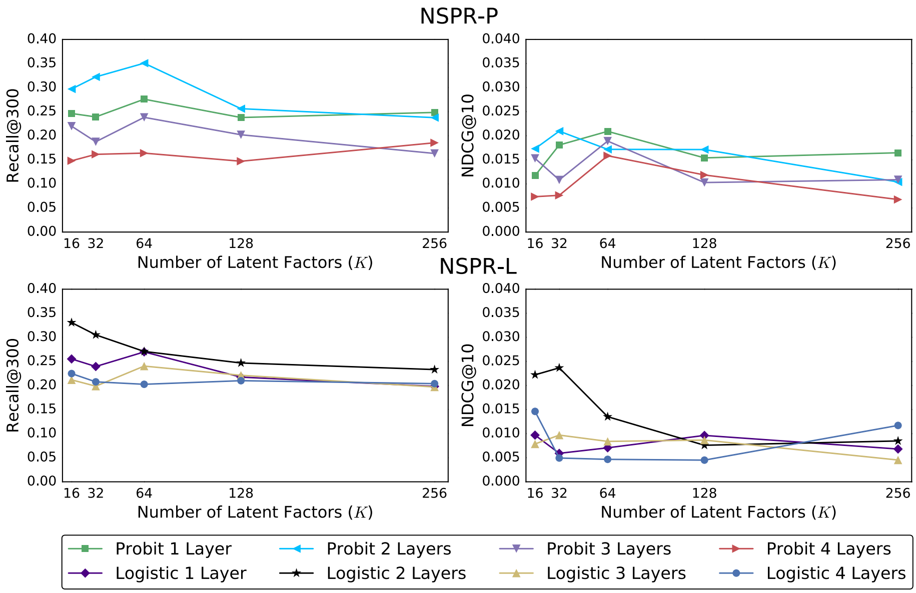

Fig. 4 Recall@300 (*left*) and NDCG@10 (*right*) for varying number of latent factors (*K*) and hidden layers (*L*) averaged over tenfolds on the *Yahoo! Movies* dataset

architecture is achievable. In some cases, high Recall does not directly translate to the NDCG metric. Particularly we observe this behavior in NSPR-L with four hidden layers and 256 latent factors, where Recall is among the lowest obtained yet NDCG is among the highest. Demonstrating NSPR is flexible, and its architecture can be fine tuned for specific metrics.

Table 4 summarizes the variance of NSPR over each of the tenfolds for Recall and NDCG@10. Similar to the *citeulike-a* dataset, the variance is generally low across different

Table 4 Variance over tenfolds of NSPR variants for varying the number of latent factors (*K*) and hidden layers (*L*) on the *Yahoo! Movies* dataset

	Recall@300					NDCG@10				
	16	32	64	128	256	16	32	64	128	256
P/1	0.0205	0.0184	0.0116	0.0140	0.0134	0.0001	0.0002	0.0001	0.0001	0.0002
P/2	0.0146	0.0060	0.0053	0.0137	0.0128	0.0001	0.0000	0.0000	0.0005	0.0001
P/3	0.0091	0.0136	0.0171	0.0096	0.0107	0.0003	0.0001	0.0005	0.0001	0.0003
P/4	0.0166	0.0089	0.0058	0.0067	0.0119	0.0000	0.0000	0.0004	0.0004	0.0001
L/1	0.0124	0.0190	0.0185	0.0135	0.0246	0.0000	0.0000	0.0000	0.0002	0.0000
L/2	0.0196	0.0071	0.0138	0.0137	0.0166	0.0004	0.0016	0.0003	0.0001	0.0001
L/3	0.0121	0.0122	0.0167	0.0137	0.0140	0.0001	0.0002	0.0001	0.0002	0.0000
L/4	0.0123	0.0138	0.0118	0.0134	0.0143	0.0007	0.0000	0.0000	0.0000	0.0005

We denote P/1 to indicate NSPR-Probit with one hidden layer and similarly, L/2 to indicate NSPR-Logistic with two hidden layers configurations. Since the diversity on NDCG is too small, we limit our discussion to the Recall metric. NSPR demonstrates the highest variance with a single hidden consisting of 16 and 256 latent factors for NSPR-P and NSPR-L respectively. Cumulatively, NSPR-L has more variance. However, the variance reduces as the number of layers increases. A similar yet more subtle trend is present in the Probit version. We could speculate the optimization of the non-convex nature causes the DNN to become stuck at a saddle point or bad local minimum producing the variance. Nevertheless, the small size of the dataset could easily lead to overfitting a large number of parameters. In this case, initializing the DNN weights with pretrained word embeddings may improve performance. In short, the *Yahoo! Movies* dataset contains denser ratings with sparse item content leading to a more complex relation where deeper architectures can capture these nonlinearlities.

4.4.2 Baseline comparison

Table 5 contains the results of NSPR compared to the baseline models measuring Recall@M, MAP@500, NDCG@5, and NDCG@10 on the *citeulike-a* dataset. We can see that both NSPR models perform equally well and outperform all baselines across all metrics. The nearest competitor is CTR for Recall@300. Concerning MAP@500 and NDCG, the three models using deep learning (NSPR-P, NSPR-L and CDL) obtain superior performance over models that do not. We can speculate deep learning methods utilize learned latent semantics from item content to prioritize more relevant items. CMF outperforms SVDFeature when the metric is at a higher level, i.e. when Recall is at 100 or greater but SVDFeature reports better NDCG while both have similar performance on MAP@500. SVDFeature may place a higher priority on relevant recommendations by drawing upon stronger user-based features. Both models use relatively simple linear transformations on the item content deteriorating performance to generalize to new items. These results indicate the benefits of using deep learning to construct robust feature representations of item content for the cold-start problem.

In the *Yahoo! Movies* dataset, NSPR models outperform or demonstrate competitive performance against each baseline for all metrics shown in Table 6. Again, NSPR-L

Table 5 Experimental results for different methods on the *citeulike-a* dataset

	SVDFeature	CMF	CTR	CDL	NSPR-P	NSPR-L
R@10	0.0039	0.0023	0.0692	0.0919	**0.1294**	0.1290
R@25	0.0095	0.0055	0.1516	0.1693	**0.2324**	0.2308
R@50	0.0188	0.0110	0.2518	0.2580	**0.3402**	0.3378
R@100	0.0335	0.0562	0.3802	0.3634	**0.4716**	0.4646
R@150	0.0493	0.0919	0.4616	0.4304	**0.5526**	0.5443
R@200	0.0666	0.1066	0.5197	0.4807	**0.6100**	0.6042
R@250	0.0825	0.1198	0.5647	0.5203	**0.6547**	0.6515
R@300	0.0985	0.1459	0.6044	0.5514	**0.6862**	0.6859
MAP@500	0.0025	0.0026	0.0522	0.0672	**0.0923**	0.0906
NDCG@5	0.0027	0.0024	0.0457	0.0773	**0.1296**	0.1259
NDCG@10	0.0039	0.0026	0.0578	0.0809	**0.1432**	0.1418

The best results in each metric are bold

Table 6 Experimental results for different methods on the *Yahoo! Movies* dataset

	SVDFeature	CMF	CTR	CDL	NSPR-P	NSPR-L
R@10	0.0042	0.0013	0.0051	0.0234	0.0200	**0.0453**
R@25	0.0109	0.0040	0.0112	0.0414	0.1193	**0.1361**
R@50	0.0209	0.0090	0.0200	0.0653	0.0619	**0.0840**
R@100	0.0427	0.0324	0.0336	0.1071	0.2054	**0.2127**
R@150	0.0625	0.0769	0.0495	0.1439	0.2764	**0.3010**
R@200	0.0837	0.1161	0.0639	0.1816	0.3377	**0.3559**
R@250	0.1046	0.1395	0.0778	0.2181	0.4436	**0.4437**
R@300	0.1259	0.1551	0.0903	0.2518	0.5179	**0.5266**
MAP@500	0.0034	0.0022	0.0042	0.0168	0.0173	**0.0221**
NDCG@5	0.0028	0.0015	0.0044	0.0172	0.0094	**0.0217**
NDCG@10	0.0035	0.0018	0.0046	0.0175	0.0186	**0.0380**

The best results in each metric are bold

performs best with NSPR-P performing very closely. As noted earlier, the dataset is characterized by denser ratings and sparser item content may lead to a more complex relation. Subsequently, topic models may lack the ability to capture this intricate relationship with sparser documents leading to CTR's poor performance. SVDFeature and CMF both obtain better Recall@300 at 0.1259 and 0.1551, respectively. The fact that CDL outperforms other baselines additionally with NSPR's performance demonstrate the advantages of deep learning models which aim to capture complex and subtle relations between item content and latent features. The NSPR framework's flexibility to integrate different types of pairwise probability functions demonstrates its adaptability. Furthermore, the difference in NSPR variations performance is the pairwise function. The Probit function's hyperparameters μ and σ^2 could be further optimized to suit different dataset characteristics which we leave to future work. These results prove the effectiveness of NSPR using a pairwise probability for implicit feedback and utilizing DNN for learning latent semantics from item content, compared to the pointwise loss and auto-encoder in CDL. As we can see, the NSPR models demonstrate competitive or superior performance over the state-of-the-art baselines across all metrics.

4.4.3 Architecture of NSPR

In this section, we more closely examine the architecture of NSPR by varying the number of hidden layers from $L = 1, 2, 3, 4$ using the default parameters. Table 7 reports the

Table 7 Recall@300 for NSPR with different number of hidden layers ($L = 1, 2, 3, 4$) for *citeulike-a*

Hidden layers (L)	*citeulike-a*			
	1	2	3	4
NSPR-P	0.6730	0.6862	0.6663	0.6674
NSPR-L	0.5696	0.6859	0.6638	0.6381

Table 8 Recall@300 for NSPR with different number of hidden layers ($L = 1, 2, 3, 4$) for *Yahoo! Movies*

Hidden layers (L)	*Yahoo! Movies*			
	1	2	3	4
NSPR-P	0.4583	0.5179	0.3921	0.1414
NSPR-L	0.4548	0.5266	0.4393	0.0941

values of the *citeulike-a* dataset. We can see the performance is relatively stable across different number of hidden layers for NSPR-P. For NSPR-L, we can see a single hidden layer does not provide enough modeling capacity as performance increases with additional layers. Both models obtain the best results with two hidden layers and increasing the number of layers may result in overfitting. In the *Yahoo! Movies* dataset, we see performance also peaking around two hidden layers and then sharply decreasing at four hidden layers in Table 8. We believe this to be an overfitting issue from the DNN, also observed by Salakhutdinov et al. (2007). Dropout lead to a decrease in performance and subsequently decreased the stability of the method across the number of hidden layers in contrast to the previous results on the *citeulike-a* dataset. In general, a single hidden layer architecture lacks the modeling capacity to capture these nonlinearlities where the two hidden layer architecture excelled. Additional layers lead to overfitting possibly due to the small size of the dataset. In future work, we plan to apply deeper architectures with large-scale test beds and exploit external knowledge such as pretrained word embeddings.

In the core architecture of NSPR, the DNN is approximating the item latent space and not a directly observable variable. One could view the item latent factor as an additional hidden layer connecting to the DNN. The initial error propagates to the latent item vector then we evaluate another error function with respect to the DNN output. We experimented with different combinations of activation functions ranging from the tanh, Logistic, Rectified Linear Unit (ReLU) (Nair and Hinton 2010) and identity. We found the best performance with the tanh function and the identity as the output. The Logistic function provided slightly deteriorated performance. One explanation may be the Logistic function is bound from [0, 1] slowing learning by outputting a positive mean as inputs to subsequent hidden layers whereas the symmetry of tanh generally provides a zero centered mean typically leading to better convergence (LeCun et al. 2012). In our particular problem, we did not find ReLU's to enhance performance as demonstrated in Cheng et al. (2016).

4.4.4 *Impact of item variance*

The item variance σ_v^2 in Eq. (2) models the interaction between the semantic learning of DNN from item content and latent factor learning from implicit feedback. In this section, we investigate the impact of σ_v^2 on the NSPR models and vary it from the default values specified in Sect. 4.2. We vary σ_v^2 by ± 5 of our default values followed by more extraneous values. Table 9 demonstrates the effect of σ_v^2 on the *citeulike-a* dataset. As we can see, NSPR's performance is relatively robust over a broad range of values and generally, shows relatively subtle changes with the exception when σ_v^2 is small i.e. 0.1. In contrast, the *Yahoo! Movies* dataset shows more sensitivity to σ_v^2 in performance as shown in Table 10.

Table 9 Recall@300 for different values of σ_v^2 on the *citeulike-a* dataset

σ_v^2	0.1	495	500	505	1000
NSPR-P	0.0875	0.6794	0.6862	0.6812	0.6626
NSPR-L	0.0802	0.6763	0.6859	0.6474	0.6410

Table 10 Recall@300 for different values of σ_v^2 on the *Yahoo! Movies* dataset

σ_v^2	0.1	195	200	205	250
NSPR-P	0.2619	0.4211	0.5179	0.4475	0.4696
σ_v^2	0.1	4	9	13	100
NSPR-L	0.2889	0.4125	0.5266	0.4833	0.4873

When σ_v^2 is small, the DNN strongly influencing the item latent factor overfitting to item content. As the value of σ_v^2 increases, the DNN item content integration starts to diverge from the item latent factor and as σ_v^2 goes to infinity the model degenerates to the BPR criterion. These results demonstrate the importance of keeping a balance between pairwise probability and latent semantic learning of DNN.

4.5 Qualitative evaluation

To further investigate the effectiveness of NSPR, we compare the interpretability of the top 5 recommended items against baselines for a given user. Table 11 lists the recommended articles for *citeulike-a* by NSPR-L, CDL, and CTR. We might hypothesize that this user is interested in library and information sciences. CTR correctly recommends only two articles while four out of the five article titles recommended contain the root word ‘science.’ The remaining article is ‘In a paperless world a new role for academic libraries: providing open access’ which we can also expect the terms ‘academic’ and ‘library’ to co-occur with ‘science’ leading CTR astray. Similarly, CDL identified words ‘technology’ and ‘publication’ while correctly recommending one item. CDL incorrectly recommends the article ‘The Molecular Biology Database Collection: 2005 update’. Upon inspecting the training data, the user does not have any interests in biology. CDL may have identified ‘database’ as a term co-occurring with ‘digital’, ‘libraries’ and ‘publications.’ NSPR-L captures more semantics related to the users primary interests such as digital library and citation metrics.

The top 5 recommended movies from the *Yahoo! Movies* dataset is listed in Table 12. Analyzing the genres of each movie recommended we may speculate the user has diverse tastes in movies from a variety of genres spanning comedy, action, and adventure. CTR and CDL do not identify the action and adventure genre which comprises a significant portion of the user’s preferences while NSPR-L discovered the association, particularly in recommending ‘Indiana Jones and the Last Crusade.’ It may seem odd that NSPR-L recommended the horror films ‘Bloody Murder’, however, inspecting the users library revealed additional horror movies such as ‘Texas Chainsaw Massacre.’

Table 11 Top-5 recommended articles by NSPR-L, CDL and CTR

NSPR-L
1. Ten-Year Cross-Disciplinary Comparison of the Growth of Open Access and How it Increases Research Citation Impact
2. Peer Review in the Google Age: Is technology changing the way science is done and evaluated?
3. Defrosting the digital library: bibliographic tools for the next generation web
4. What are digital libraries? Competing visions
5. A New Era in Citation and Bibliometric Analyses: Web of Science, Scopus, and Google Scholar
CTR
1. Do pressures to publish increase scientists' bias? An empirical support from US states data
2. Unavailability of online supplementary scientific information from articles published in major journals
3. Strategic reading, ontologies, and the future of scientific publishing
4. In a paperless world a new role for academic libraries: providing open access
5. Universality of citation distributions: Toward an objective measure of scientific impact
CDL
1. Where do educational technologists really publish? An examination of successful emerging scholars' publication outlets
2. The Molecular Biology Database Collection: 2005 update
3. Strategic reading, ontologies, and the future of scientific publishing
4. Peer Review in the Google Age: Is technology changing the way science is done and evaluated?
5. Déjà? vu—a study of duplicate citations in Medline

The positive items are bold

Table 12 Top-5 recommended movies by NSPR-L, CDL and CTR

NSPR-L
1. American Wedding (2003)
2. Tarzan and the Lost City (1998)
3. Indiana Jones and the Last Crusade (1989)
4. Bloody Murder (1999)
5. Bloody Murder 2 (2003)
CDL
1. Virus (1980)
2. Take This Job and Shove It (1981)
3. Going Greek (2001)
4. Shine (1997)
5. Bless the Beasts and Children (1972)
CTR
1. Ricochet River (2001)
2. Buffalo Soldiers (1988)
3. Tempest (1982)
4. Two Hands (1999)
5. JFK (1991)

The positive items are bold

5 Conclusion and future work

Item cold-start and implicit user feedback present two of the greatest challenges to the real-world recommender systems. In this paper, we tackle the challenges by proposing a novel probabilistic generative modeling approach to integrate deep neural network (DNN) with three pairwise ranking variants. With the modeling power of deep learning, we can extract semantic representation of items and couple it with the latent factors learned from implicit feedback. The experiments show that the proposed approach significantly outperforms the competitive baselines on two real-world public datasets.

This work is just an initial step towards a promising new direction. In future work, we plan to incorporate other types of deep learning architectures such as CNN (LeCun et al. 1998), Deep belief network (DBN) (Hinton et al. 2006), and Recurrent neural network (RNN) (Bengio et al. 2003). Further performance boost may be possible when using such deep learning models since these models can explicitly take the context and ordering of words into account. Moreover, we plan to explore deeper architectures by applying data normalization techniques (Ioffe and Szegedy 2015; Ba et al. 2016) to help model stability and a better local optimum. Last but not the least, the proposed NSPR framework can be readily extended to handle the listwise preferences if ranked lists of items are given as ground truth for recommendations. The listwise learning to rank likelihood functions such as ListMLE and ListNet (Liu 2009) can be directly plugged into the proposed generative framework. The listwise approach may be able to handle more complex user feedback than pairwise preferences.

References

Ba, J. L., Kiros, J. R., & Hinton, G. E. (2016). *Layer normalization*. arXiv preprint arXiv:160706450.

Bengio, Y., Ducharme, R., Vincent, P., & Jauvin, C. (2003). A neural probabilistic language model. *Journal of Machine Learning Research*, *3*(Feb), 1137–1155.

Bennett, J., & Lanning, S. (2007). The netflix prize. In *SIGKDD Cup* (Vol. 2007, p. 35).

Bottou, L. (2010). Large-scale machine learning with stochastic gradient descent. In *COMPSTAT* (pp. 177–186). Berlin: Springer.

Chen, T., Zhang, W., Lu, Q., Chen, K., Zheng, Z., & Yu, Y. (2012). Svdfeature: A toolkit for feature-based collaborative filtering. *Journal of Machine Learning Research*, *13*(1), 3619–3622.

Cheng, C., Yang, H., King, I., & Lyu, M. R. (2012). Fused matrix factorization with geographical and social influence in location-based social networks. In *AAAI*.

Cheng, H. T., Koc, L., Harmsen, J., Shaked, T., Chandra, T., Aradhye, H., Anderson, G., Corrado, G., Chai, W., Ispir, M. et al. (2016). *Wide & deep learning for recommender systems*. arXiv preprint arXiv:160607792

Deng, L., & Yu, D. (2014). Deep learning: Methods and applications. *Foundations and Trends in Signal Processing*, *7*(3–4), 197–387.

Duchi, J., Hazan, E., & Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, *12*, 2121–2159.

Dziugaite, G. K., & Roy, D. M. (2015). *Neural network matrix factorization*. CoRR arXiv:1511.06443.

Gantner, Z., Drumond, L., Freudenthaler, C., & Schmidt-Thieme, L. (2012). Bayesian personalized ranking for non-uniformly sampled items. *Journal of Machine Learning Research*, *18*, 231–247.

Georgiev, K., & Nakov, P. (2013). A non-iid framework for collaborative filtering with restricted boltzmann machines. In *ICML* (pp. 1148–1156).

He, X., Liao, L., Zhang, H., Nie, L., Hu, X., & Chua, T. S. (2017). Neural collaborative filtering. In *Proceedings of the 26th international world wide web conference*.

Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A. R., Jaitly, N., et al. (2012). Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, *29*(6), 82–97.

Hinton, G. E., Osindero, S., & Teh, Y. W. (2006). A fast learning algorithm for deep belief nets. *Neural Computation*, *18*(7), 1527–1554.

Hu, Y., Koren, Y., & Volinsky, C. (2008). Collaborative filtering for implicit feedback datasets. In *ICDM* (pp. 263–272). IEEE.

Ioffe, S., & Szegedy, C. (2015). *Batch normalization: Accelerating deep network training by reducing internal covariate shift*. arXiv preprint arXiv:150203167

Jing, H., & Smola, A. J. (2017). Neural survival recommender. In *Proceedings of the tenth ACM international conference on web search and data mining (WSDM)* (pp. 515–524). New York, NY: ACM.

Koren, Y. (2010). Factor in the neighbors: Scalable and accurate collaborative filtering. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, *4*(1), 1.

Krohn-Grimberghe, A., Drumond, L., Freudenthaler, C., & Schmidt-Thieme, L. (2012). Multi-relational matrix factorization using bayesian personalized ranking for social network data. In *WSDM* (pp. 173–182). ACM.

LeCun, Y. A., Bottou, L., Orr, G. B., & Müller, K. R. (2012). Efficient backprop. In *Neural networks: Tricks of the trade* (pp. 9–48). Berlin: Springer.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324.

Levy, O., & Goldberg, Y. (2014). Neural word embedding as implicit matrix factorization. In *Advances in neural information processing systems* (pp. 2177–2185).

Li, S., Kawale, J., & Fu, Y. (2015). Deep collaborative filtering via marginalized denoising auto-encoder. In *CIKM* (pp. 811–820). ACM.

Liang, D., Altosaar, J., Charlin, L., & Blei, D. M. (2016). Factorization meets the item embedding: Regularizing matrix factorization with item co-occurrence. In *Proceedings of the 10th ACM conference on recommender systems* (pp. 59–66). ACM.

Linden, G., Smith, B., & York, J. (2003). Amazon.com recommendations: Item-to-item collaborative filtering. *Internet Computing*, *7*(1), 76–80.

Liu, T. Y. (2009). Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, *3*(3), 225–331.

Manning, C. D., Raghavan, P., Schütze, H., et al. (2008). *Introduction to information retrieval*. Cambridge: Cambridge University Press.

McCullagh, P., & Nelder, J. A. (1989). *Generalized linear models* (Vol. 37). Boca Raton, FL: CRC Press.

Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)* (pp. 807–814).

Pan, R., Zhou, Y., Cao, B., Liu, N. N., Lukose, R., Scholz, M., & Yang, Q. (2008). One-class collaborative filtering. In *ICDM* (pp. 502–511). IEEE.

Pan, W., & Chen, L. (2013). GBPR: Group preference based Bayesian personalized ranking for one-class collaborative filtering. In *IJCAI* (Vol. *13*, pp. 2691–2697).

Pazzani, M. J., & Billsus, D. (2007). Content-based recommendation systems. In *The adaptive web* (pp. 325–341). Berlin: Springer.

Rendle, S., & Freudenthaler, C. (2014). Improving pairwise learning for item recommendation from implicit feedback. In *WSDM* (pp. 273–282). New York, NY: ACM Press.

Rendle, S., Freudenthaler, C., Gantner, Z., & Schmidt-thieme, L. (2009). BPR: Bayesian personalized ranking from implicit feedback. In *UAI* (pp. 452–461).

Rendle, S., & Schmidt-Thieme, L. (2010). Pairwise interaction tensor factorization for personalized tag recommendation. In *WSDM* (pp. 81–90). ACM.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1988). Learning representations by back-propagating errors. *Cognitive Modeling*, *5*(3), 1.

Salakhutdinov, R., Mnih, A., & Hinton, G. (2007). Restricted Boltzmann machines for collaborative filtering. In *ICML* (pp. 791–798). ACM.

Sedhain, S., Menon, A. K., Sanner, S., & Xie, L. (2015). AutoRec: Autoencoders meet collaborative filtering. In *WWW* (pp. 111–112).

Singh, A. P., & Gordon, G. J. (2008). Relational learning via collective matrix factorization. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 650–658). ACM.

Strub, F., & Jeremie, M. (2015). Collaborative filtering with stacked denoising AutoEncoders and sparse inputs. In *NIPS workshop on machine learning for eCommerce*. Montreal

Van den Oord, A., Dieleman, S., & Schrauwen, B. (2013). Deep content-based music recommendation. In *NIPS* (pp. 2643–2651).

Wang, C., & Blei, D. M. (2011). Collaborative topic modeling for recommending scientific articles. In *SIGKDD* (pp. 448–456).

Wang, H., Wang, N., & Yeung, D. Y. (2015). Collaborative deep learning for recommender systems. In *SIGKDD*.

Wang, X., & Wang, Y. (2014). Improving content-based and hybrid music recommendation using deep learning. In: *International conference on multimedia* (pp. 627–636). ACM

Wu, C. Y., Ahmed, A., Beutel, A., Smola, A. J., & Jing, H. (2017). Recurrent recommender networks. In *Proceedings of the tenth ACM international conference on web search and data mining (WSDM)* (pp. 495–503). New York, NY: ACM.

Wu, Y., Dubois, C., Zheng, A. X., & Ester, M. (2016). Collaborative denoising auto-encoders for Top-N recommender systems. In *WSDM*.

Ying, H., Chen, L., Xiong, Y., & Wu, J. (2016). Collaborative deep ranking: A hybrid pair-wise recommendation algorithm with implicit feedback. In *PAKDD*.

Zhang, F., Yuan, N. J., Lian, D., Xie, X., & Ma, W. Y. (2016a). Collaborative knowledge base embedding for recommender systems. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 353–362). ACM.

Zhang, W., Du, T., & Wang, J. (2016b). Deep learning over multi-field categorical data. In *European conference on information retrieval* (pp. 45–57). Berlin: Springer.

Zheng, L., Noroozi, V., & Yu, P. S. (2017). Joint deep modeling of users and items using reviews for recommendation. In *Proceedings of the tenth ACM international conference on web search and data mining, WSDM '17* (pp. 425–434). New York, NY: ACM. doi:10.1145/3018661.3018665.