On strong convergence of the two-tower model for recommender system

Anonymous authors
Paper under double-blind review

ABSTRACT

Recommender system is capable of predicting preferred items for a user by integrating information from similar users or items. A popular model in recommender system is the so-called two-tower model, which employs two deep neural networks to embed users and items into a low-dimensional space, and predicts ratings via the geometrical relationship of the embeddings of user and item in the embedded space. Even though it is popularly used for recommendations, its theoretical properties remain largely unknown. In this paper, we establish some asymptotic results of the two-tower model in terms of its strong convergence to the optimal recommender system, showing that it achieves a fast convergence rate depending on the intrinsic dimensions of inputs features. To the best of our knowledge, this is among the first attempts to establish the statistical guarantee of the two-tower model. Through numerical experiments, we also demonstrate that the two-tower model is capable of capturing the effects of users' and items' features on ratings, leading to higher prediction accuracy over its competitors in both simulated examples and a real application data set.

Keywords: Artificial neural networks, Collaborative filtering, Empirical process, Recommender system, Two-tower model

1 INTRODUCTION

Recommender system has attracted great interest in machine learning community in the past two decades, mostly due to its wide applications in e-commerce and precision marketing, such as movie recommendation (Miller et al., 2003), news feeding (Li et al., 2016), online shopping (Romadhony et al., 2013) and restaurant selection (Vargas-Govea et al., 2011). Existing recommender systems can be broadly categorized into three categories: content-based filtering (Boratto et al., 2017; Lang, 1995), collaborative filtering (Koren, 2009; Hofmann and Puzicha, 1999), and hybrid methods (Gunawardana and Meek, 2009). Content-based filtering methods entail preprocessing techniques to transform unstructured item contents and user profiles into numerical vectors, which are used as inputs for classical machine learning algorithms, such as decision tree (Middleton et al., 2004), SVM (Fortuna et al., 2010; Oku et al., 2006), and kNN (Subramaniyaswamy and Logesh, 2017). Collaborative filtering methods predicts user's ratings based on other similar users or similar items, including singular value decomposition (SVD; Mazumder et al. 2010), restricted Boltzman machines (RBM; Salakhutdinov et al. 2007), probabilistic latent semantic analysis (Hofmann, 2004) and nearest neighbors methods (Koren et al., 2009). Hybrid recommender systems seek out to combine collaborative filtering and content-based filtering, including the unified Boltzmann machines (Gunawardana and Meek, 2009) and partial latent vector model (Zhu et al., 2016). Gunawardana and Meek (2009) proposed a unified Boltzmann machines to encode collaborative and content information as features for predicting ratings. Zhu et al. (2016) integrated additional user-specific and content-specific predictors in partial latent vector model in an additive fashion to achieve a better prediction accuracy.

In recent years, artificial neural network has been popularly employed in recommender system, and success has been widely reported in various applications. One of the most popular neural network model for recommender system is the so-called two-tower model (Yi et al., 2019), where two deep neural networks, referred to as towers, act as encoders to embed high-dimensional users' and items' features into a low-dimensional space. A key advantage of the two-tower model is that it is able to tackle the long-standing cold-start issue by incorporating users' and items' features to produce accurate recommendations for brand new users or items. In literature, the two-tower model has been widely employed in various applications, including video recommendation (Wang et al., 2021a; Yi et al., 2019), application recommendation (Yang et al., 2020), book recommendation (Wang et al., 2021b), yet its theoretical foundation is still largely unavailable.

The main contribution of this paper is to establish some asymptotic properties of the two-tower model in terms of its strong convergence to the optimal recommender system. Specifically, under the assumption that each embedding dimension of user or item is a smooth function of the corresponding input features, we conduct an error analysis of the approximation and estimation errors of the two-tower model. It can be showed that the strong convergence of the two-tower model largely depends on the smoothness of the optimal recommender system as well as the intrinsic dimension of user and item features. Its rate of convergence becomes faster as the underlying smoothness of the true model increases or the maximum intrinsic dimensions of user and item features decrease. Particularly, when the underlying smoothness goes to infinity, the convergence rate of the two-tower model becomes $O_p(|\Omega|^{-1}(\log|\Omega|)^2)$ with Ω denoting the observed rating set and $|\cdot|$ denoting the set cardinality, which is faster than most existing theoretical results in literature Zhu et al. (2016). Most importantly, the established statistical guarantee for the two-tower model provides some solid theoretical justification for the success of the two-tower model in various applications.

The rest of this paper is structured as follows. Section 1.1 introduces the notations and definitions to be used in the sequel. Section 2 introduces the framework of the two-tower model. Section 3 establishes the asymptotic properties of the two-tower model with respect to its approximation error and strong convergence. Section 4 conducts a variety of numerical experiments and real applications to demonstrate the advantage of the two-tower model. A brief summary is provided in Section 5, and all technical proofs are contained in a separate supplementary file.

1.1 Notations and definitions

For a probability measure μ, its support is denoted as $\mathrm{Supp}(\mu)$. For a function $g:\mathbb{R}^D\to\mathbb{R}$, its $L^2(\mu)$-norm and $L^\infty(\mu)$-norm with respect to μ are $\|g\|_{L^2(\mu)}=\left(\int_{\boldsymbol{x}} g^2(\boldsymbol{x})d\mu(\boldsymbol{x})\right)^{1/2}$ and $\|g\|_{L^\infty(\mu)}=\sup_{\boldsymbol{x}\in\mathrm{Supp}(\mu)} g(\boldsymbol{x})$, respectively. For a vector $\boldsymbol{x}$, its l_2-norm is $\|\boldsymbol{x}\|_2=(\sum_{i=1}^p x_i^2)^{1/2}$. For a set S, we define $\mathcal{N}(\epsilon,S,\|\cdot\|)$ as the minimal number of ϵ-balls needed to cover S under a generic metric $\|\cdot\|$.

Let $\boldsymbol{f}$ be an L-layer neural network,

$$\boldsymbol{f}(\boldsymbol{x};\Theta)=\boldsymbol{h}_L\circ\boldsymbol{h}_{L-1}\circ\cdots\boldsymbol{h}_1(\boldsymbol{x}),$$

where $\Theta=\big((\boldsymbol{A}_1,\boldsymbol{b}_1),\ldots,(\boldsymbol{A}_L,\boldsymbol{b}_L)\big)$ denotes all the parameters, $\circ$ denotes function composition, and $\boldsymbol{h}_l(\boldsymbol{x})=\sigma(\boldsymbol{A}_l\boldsymbol{x}+\boldsymbol{b}_l)$ denotes the l-th layer. Here $\boldsymbol{A}_l\in\mathbb{R}^{p_l\times p_{l-1}}$ is the weight matrix, $\boldsymbol{b}_l\in\mathbb{R}^{p_l}$ is the bias term, p_l is the number of neurons in the l-th layer, and $\sigma(\cdot)$ is a component-wise activation function, such as the sigmoid function $\sigma(x)=1/(1+\exp(-x))$, or the ReLU function $\sigma(x)=\max(x,0)$. For ease of notation, $\boldsymbol{f}(\boldsymbol{x};\Theta)$ will be abbreviated as $\boldsymbol{f}(\boldsymbol{x})$ when it causes no confusion in the sequel. To characterize the network architecture of $\boldsymbol{f}$, we denote its number of layers as $U(\boldsymbol{f})$, its scale of parameters as $D(\boldsymbol{f})=\max_{l=1,\ldots,U(\boldsymbol{f})}\max\{\|\boldsymbol{b}_l\|_\infty,\|\mathrm{vec}(\boldsymbol{A}_l)\|_\infty\}$, and its number of effective parameters as

$$Z(\boldsymbol{f})=\sum_{l=1}^{U(\boldsymbol{f})}\big(\|\boldsymbol{b}_l\|_0+\|\mathrm{vec}(\boldsymbol{A}_l)\|_0\big),$$

where $\mathrm{vec}(\cdot)$ denotes the vectorization of a matrix.

Let $\beta>0$ be a degree of smoothness, then the Hölder space is defined as

$$\mathcal{H}(\beta,[0,1]^D)=\{f\in C^{\lfloor\beta\rfloor}([0,1]^D)\,|\,\|f\|_{\mathcal{H}(\beta,[0,1]^D)}<\infty\},$$

where $C^{\lfloor\beta\rfloor}([0,1]^D)$ contains all $\lfloor\beta\rfloor$-times continuously differentiable functions on $[0,1]^D$, $\lfloor\cdot\rfloor$ is the floor function, and the Hölder norm is defined as

$$\|f\|_{\mathcal{H}(\beta,[0,1]^D)}=\max_{\boldsymbol{\alpha}:\|\boldsymbol{\alpha}\|_1<\beta}\sup_{\boldsymbol{x}\in[0,1]^D}|\partial^{\boldsymbol{\alpha}}f(\boldsymbol{x})|+\max_{\boldsymbol{\alpha}:\|\boldsymbol{\alpha}\|_1=\lfloor\beta\rfloor}\sup_{\boldsymbol{x},\boldsymbol{x}'\in[0,1]^D,\boldsymbol{x}\neq\boldsymbol{x}'}\frac{|\partial^{\boldsymbol{\alpha}}f(\boldsymbol{x})-\partial^{\boldsymbol{\alpha}}f(\boldsymbol{x}')|}{\|\boldsymbol{x}-\boldsymbol{x}'\|_\infty^{\beta-\lfloor\beta\rfloor}},$$

with $\partial^{\boldsymbol{\alpha}}f=\partial_1^{\alpha_1}\cdots\partial_D^{\alpha_D}f$, $\boldsymbol{\alpha}=(\alpha_1,\ldots,\alpha_D)$, and $\alpha_i\geq 0$ is an integer. Further, we let $\mathcal{H}(\beta,[0,1]^D,M)=\{f\in\mathcal{H}(\beta,[0,1]^D)\,|\,\|f\|_{\mathcal{H}(\beta,[0,1]^D)}\leq M\}$ be a closed ball with radius M, and $\mathcal{H}^p(\beta,[0,1]^D,M)=\mathcal{H}(\beta,[0,1]^D,M)\times\mathcal{H}(\beta,[0,1]^D,M)\times\cdots\times\mathcal{H}(\beta,[0,1]^D,M)$.

2 Two-tower model

Covariates in many recommender system are unstructured and high-dimensional, such as user profiles and textual item description. It is generally believed that such information often has a low-dimensional intrinsic representation, and can be naturally integrated in the feature engineering phase in a deep learning model. Given a typical recommender system with user covariates $\boldsymbol{x}_u \in \mathbb{R}^{D_u}$ and item covariates $\tilde{\boldsymbol{x}}_i \in \mathbb{R}^{D_i}$, the two-tower model can be written as

$$R(\boldsymbol{x}_u, \tilde{\boldsymbol{x}}_i) = \langle \boldsymbol{f}(\boldsymbol{x}_u), \tilde{\boldsymbol{f}}(\tilde{\boldsymbol{x}}_i) \rangle, \tag{1}$$

where $\boldsymbol{f} : \mathbb{R}^{D_u} \to \mathbb{R}^p$ and $\tilde{\boldsymbol{f}} : \mathbb{R}^{D_i} \to \mathbb{R}^p$ are two deep neural networks mapping $\boldsymbol{x}_u$ and $\tilde{\boldsymbol{x}}_i$ into the same p-dimensional embedded space. The recommendation mechanism of the two-tower model is based on the dot product of $\boldsymbol{f}(\boldsymbol{x}_u)$ and $\tilde{\boldsymbol{f}}(\tilde{\boldsymbol{x}}_i)$, as illustrated in Figure 1.

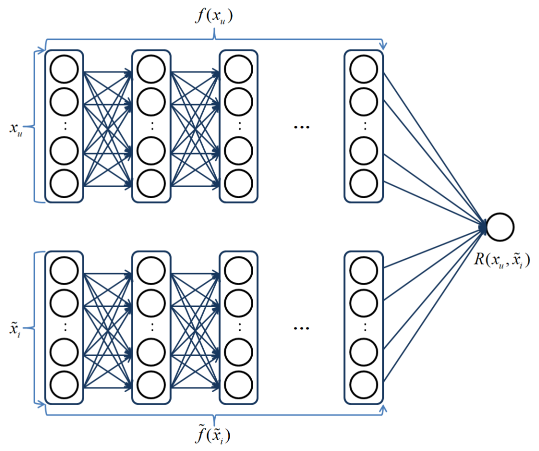

Figure 1: The neural network structure of the two-tower model.

The cost function of optimizing the two-tower model can be organized as

$$\min_{f,\tilde{f}} \frac{1}{|\Omega|} \sum_{(u,i)\in\Omega} \left(k_{ui} - \langle \boldsymbol{f}(\boldsymbol{x}_u), \tilde{\boldsymbol{f}}(\tilde{\boldsymbol{x}}_i) \rangle\right)^2 + \lambda\{J(\boldsymbol{f}) + J(\tilde{\boldsymbol{f}})\}, \tag{2}$$

where $J(\cdot)$ can be the L_1-norm or L_2-norm penalty for preventing the deep neural network from over-fitting. Moreover, it is interesting to note that (2) reduces to the classical SVD-based collaborative filtering method, when $\boldsymbol{x}_u$ and $\tilde{\boldsymbol{x}}_i$ contain only one-hot encodings for users and items.

The optimization task in (2) can be implemented via some well developed neural network computing library, such as Tensorflow (Abadi et al., 2015), which is an open-source library for large-scale machine learning algorithms. A popular scheme is to employ the stochastic gradient descent to update parameters of $\boldsymbol{f}(\boldsymbol{x}_u)$ and $\tilde{\boldsymbol{f}}(\tilde{\boldsymbol{x}}_i)$ in parallel as follows,

$$A_{ljk}^{(t+1)} = A_{ljk}^{(t)} + \frac{\alpha}{|\mathcal{M}|} \sum_{(u,i)\in\mathcal{M}} \left(k_{ui} - \langle \boldsymbol{f}(\boldsymbol{x}_u), \tilde{\boldsymbol{f}}(\tilde{\boldsymbol{x}}_i) \rangle\right) \langle \frac{d}{dA_{ljk}} \boldsymbol{f}(\boldsymbol{x}_u), \tilde{\boldsymbol{f}}(\tilde{\boldsymbol{x}}_i) \rangle - \alpha\lambda \frac{dJ(\boldsymbol{f})}{dA_{ljk}},$$

$$\tilde{A}_{ljk}^{(t+1)} = \tilde{A}_{ljk}^{(t)} + \frac{\alpha}{|\mathcal{M}|} \sum_{(u,i)\in\mathcal{M}} \left(k_{ui} - \langle \boldsymbol{f}(\boldsymbol{x}_u), \tilde{\boldsymbol{f}}(\tilde{\boldsymbol{x}}_i) \rangle\right) \langle \boldsymbol{f}(\boldsymbol{x}_u), \frac{d}{d\tilde{A}_{ljk}} \tilde{\boldsymbol{f}}(\tilde{\boldsymbol{x}}_i) \rangle - \alpha\lambda \frac{dJ(\tilde{\boldsymbol{f}})}{d\tilde{A}_{ljk}},$$

$$b_{lj}^{(t+1)} = b_{lj}^{(t)} + \frac{\alpha}{|\mathcal{M}|} \sum_{(u,i)\in\mathcal{M}} \left(k_{ui} - \langle \boldsymbol{f}(\boldsymbol{x}_u), \tilde{\boldsymbol{f}}(\tilde{\boldsymbol{x}}_i) \rangle\right) \langle \boldsymbol{f}(\boldsymbol{x}_u), \frac{d}{db_{lj}} \tilde{\boldsymbol{f}}(\tilde{\boldsymbol{x}}_i) \rangle,$$

$$\tilde{b}_{lj}^{(t+1)} = \tilde{b}_{lj}^{(t)} + \frac{\alpha}{|\mathcal{M}|} \sum_{(u,i)\in\mathcal{M}} \left(k_{ui} - \langle \boldsymbol{f}(\boldsymbol{x}_u), \tilde{\boldsymbol{f}}(\tilde{\boldsymbol{x}}_i) \rangle\right) \langle \boldsymbol{f}(\boldsymbol{x}_u), \frac{d\tilde{\boldsymbol{f}}}{d\tilde{b}_{lj}} (\tilde{\boldsymbol{x}}_i) \rangle,$$

where α is learning rate and $\mathcal{M}$ is a subset sampled uniformly from Ω. Even though the optimization task in (2) is non-convex, the algorithm is guaranteed to converge to some stationary point (Chen et al., 2012).

Compared with the classical collaborative filtering methods, the two-tower model is essentially a hybrid system by leveraging collaborative filtering and content-based filtering through the low-dimensional representations for users and items. The deep neural network structure allows for flexible representation of users and items, and thus can fully capture nonlinear covariate effect compared with linear modeling in literature (Bi et al., 2017; Mao et al., 2019). More importantly, the two-tower model can greatly alleviate the cold-start issue, by embedding new users and items via covariate representation (Van den Oord et al., 2013). It is also interesting to point out that the formulation in (2) provides a general framework for constructing deep recommender system. Modification can be carried out on the two neural network structures to adapt to various data sources, such as convolution neural network (CNN) for image data (Yang et al., 2019; Yu et al., 2019) and recurrent neural network (RNN) for sequential data (Twardowski, 2016).

3 Statistical guarantees

This section establishes some theoretical properties of the two-tower model in terms of its strong convergence to the true model, which, to the best of our knowledge, is among the first attempts to quantify the asymptotic behaviors of deep recommender systems.

We assume that the observed data $\{(\boldsymbol{x}_u, \tilde{\boldsymbol{x}}_i, k_{ui}), (u,i) \in \Omega\}$ are generated from the following model,

$$k_{ui} = R^*(\boldsymbol{x}_u, \tilde{\boldsymbol{x}}_i) + \epsilon_{ui} = \langle \boldsymbol{f}^*(\boldsymbol{x}_u), \tilde{\boldsymbol{f}}^*(\tilde{\boldsymbol{x}}_i) \rangle + \epsilon_{ui}, \tag{3}$$

where $\boldsymbol{x}_u \in [0,1]^{D_u}$, $\tilde{\boldsymbol{x}}_i \in [0,1]^{D_i}$, ϵ_{ui} are independently and identically distributed as a sub-Gaussian noise bounded by B_e with variance σ^2, and $\boldsymbol{f}^* = (f_1^*, \ldots, f_p^*)$ and $\tilde{\boldsymbol{f}}^* = (\tilde{f}_1^*, \ldots, \tilde{f}_p^*)$ with $f_j^* \in \mathcal{H}(\beta, [0,1]^{D_u}, M)$ and $\tilde{f}_j^* \in \mathcal{H}(\beta, [0,1]^{D_i}, M)$. In addition, it follows from the bounded Hölder norm that $\sup_{\boldsymbol{x} \in [0,1]^{D_u}} |f_j^*(\boldsymbol{x})| \leq M$ and $\sup_{\boldsymbol{x} \in [0,1]^{D_i}} |\tilde{f}_j^*(\boldsymbol{x})| \leq M$ for $j = 1, \ldots, p$.

3.1 Approximation error

We define two classes of bounded deep neural network for user and item as

$$\mathcal{F}_{D_u}(W, L, B, M) = \{\boldsymbol{f} | Z(\boldsymbol{f}) \leq W, U(\boldsymbol{f}) \leq L, D(\boldsymbol{f}) \leq B, \sup_{\boldsymbol{x} \in [0,1]^{D_u}} \max_{j=1,\ldots,p} |f_j(\boldsymbol{x})| \leq 2M\},$$
$$\mathcal{F}_{D_i}(\tilde{W}, \tilde{L}, \tilde{B}, M) = \{\tilde{\boldsymbol{f}} | Z(\tilde{\boldsymbol{f}}) \leq \tilde{W}, U(\tilde{\boldsymbol{f}}) \leq \tilde{L}, D(\tilde{\boldsymbol{f}}) \leq \tilde{B}, \sup_{\boldsymbol{x} \in [0,1]^{D_i}} \max_{j=1,\ldots,p} |\tilde{f}_j(\boldsymbol{x})| \leq 2M\},$$

where the boundedness of $\boldsymbol{f}$ and $\tilde{\boldsymbol{f}}$ is assumed to reduce the size of the parameter space for approximation. In the following, $\mathcal{F}_{D_u}(W, L, B, M)$ and $\mathcal{F}_{D_i}(\tilde{W}, \tilde{L}, \tilde{B}, M)$ will be abbreviated as $\mathcal{F}_{D_u}$ and $\mathcal{F}_{D_i}$, respectively, when no confusion is caused.

We further define the class of deep recommender system as

$$\mathcal{R}^\Phi = \{R(\boldsymbol{x}_u, \tilde{\boldsymbol{x}}_i) = \langle \boldsymbol{f}(\boldsymbol{x}_u), \tilde{\boldsymbol{f}}(\tilde{\boldsymbol{x}}_i) \rangle | \boldsymbol{f} \in \mathcal{F}_{D_u}(W, L, B, M), \tilde{\boldsymbol{f}} \in \mathcal{F}_{D_i}(\tilde{W}, \tilde{L}, \tilde{B}, M)\},$$

where $\Phi = (W, L, B, M, \tilde{W}, \tilde{L}, \tilde{B})$ denotes the parameters determining the size of $\mathcal{R}^\Phi$. The estimate $\hat{R}$ is then defined as

$$\hat{R} = \arg\min_{R \in \mathcal{R}^\Phi} \frac{1}{|\Omega|} \sum_{(u,i) \in \Omega} \big(k_{ui} - R(\boldsymbol{x}_u, \tilde{\boldsymbol{x}}_i)\big)^2 + \lambda J(R),$$

where $J(R) = \sum_{l=1}^{L} (\|\boldsymbol{A}_l\|_F^2 + \|\boldsymbol{b}_l\|_2^2) + \sum_{l=1}^{\tilde{L}} (\|\tilde{\boldsymbol{A}}_l\|_F^2 + \|\tilde{\boldsymbol{b}}_l\|_2^2)$.

We first quantify the approximation error of the two-tower model in Theorem 1, which builds upon the theoretical treatments in Nakada and Imaizumi (2020) to accommodate specific challenges in deep recommender system. Particularly, the high-dimensional input for deep recommender system often live on a low-dimensional manifold, especially those containing sparse binarized features such as bag-of-words or one-hot encoding. To quantify the intrinsic dimension of the input space S, its upper Minkowski dimension of S (Falconer, 2004) is defined as

$$\dim(S) = \inf\{d^* \geq 0 | \limsup_{\epsilon \to 0} \mathcal{N}(\epsilon, S, \|\cdot\|_\infty)\epsilon^{d^*} = 0\}.$$

It is important to note that the upper Minkowski dimension of a discrete input space is always 0, and thus binarized features normally do not increase the upper Minkowski dimension when integrated into the input for deep recommender system.

Theorem 1. *Suppose* $\dim\big(\text{Supp}(\mu_u)\big) \le d_u$ *and* $\dim\big(\text{Supp}(\mu_i)\big) \le d_i$*, where* μ_u *and* μ_i *denote the probability measure of* $\boldsymbol{x}_u$ *and* $\tilde{\boldsymbol{x}}_i$*, respectively. Then for any* $\epsilon > 0$*, there exists* $\Phi = (W, L, B, M, \tilde{W}, \tilde{L}, \tilde{B})$ *with* $W = O(\epsilon^{-d_u/\beta})$*,* $\tilde{W} = O(\epsilon^{-d_i/\beta})$*,* $B = O(\epsilon^{-s})$ *and* $\tilde{B} = O(\epsilon^{-s})$*, such that*

$$\inf_{R \in \mathcal{R}^{\Phi}} \|R - R^*\|_{L^\infty(\mu_{ui})} \le 3pM\epsilon,$$

where μ_{ui} *denotes the probability measure of* $(\boldsymbol{x}_u, \tilde{\boldsymbol{x}}_i)$ *on* $\text{Supp}(\mu_u) \times \text{Supp}(\mu_i)$*.*

Theorem 1 quantifies the approximation error of the two-tower model. Its proof as well as the proofs for all other lemmas and theorems are provided in a separate supplementary file. The upper bound on the approximation error in Theorem 1 assures that the true model can be well approximated by $\mathcal{R}^{\Phi}$ for some Φ, as long as the underlying true functions $\boldsymbol{f}^*$ and $\tilde{\boldsymbol{f}}^*$ in (3) are sufficiently smooth. Furthermore, Theorem 1 holds true regardless of the value of L, implying that the approximation error of the two-tower model can converge to 0 with any number of layers.

3.2 Strong convergence

We are now ready to lay out some preparatory lemmas that are necessary to quantify the strong convergence of the two-tower model. Specifically, the following lemmas are established to measure the entropy of $\mathcal{R}^{\Phi}$, which plays a key role in deriving the estimation error of $\hat{R}$ and strikes a balance between its estimation and approximation errors.

Lemma 1. *Let* $S_B(c, d) = \{(\boldsymbol{A}, \boldsymbol{b}) | \boldsymbol{A} \in [-B, B]^{c \times d}, \boldsymbol{b} \in [-B, B]^c\}$*, and*

$$\mathcal{K}_D(W, L, B, M) = \{\boldsymbol{f}(\cdot; \Theta) : \Theta \in S_B(2W, D) \times S_B^{L-2}(2W, 2W) \times S_B(p, 2W))\}.$$

There exists a mapping $Q : \mathcal{F}_D(W, L, B, M) \to \mathcal{K}_D(W, L, B, M)$ *such that* $\boldsymbol{f}(\boldsymbol{x}) = Q(\boldsymbol{f})(\boldsymbol{x})$ *for any* $\boldsymbol{f} \in \mathcal{F}_D(W, L, B, M)$*, where* $Z(Q(\boldsymbol{f})) \le 14LW \log W$*.*

Note that $\mathcal{F}_{D_u}$ and $\mathcal{F}_{D_i}$ contain neural networks with different layers and widths, making it difficult to carve out analyzable forms for their entropy. Lemma 1 shows that both $\mathcal{F}_{D_u}$ and $\mathcal{F}_{D_i}$ can be embedded into some relatively larger functional spaces $\mathcal{K}_{D_u}$ and $\mathcal{K}_{D_i}$, consisting of deep neural networks of a unified size. Thus, the entropy of $\mathcal{K}_{D_u}$ and $\mathcal{K}_{D_i}$ can be computed directly as a parametric model (Zhang, 2002; Wang et al., 2016; Xu et al., 2021), providing upper bounds for those of $\mathcal{F}_{D_u}$ and $\mathcal{F}_{D_i}$, respectively. It is also important to point out that except for a negligible logarithmic term, the number of effective parameters in $\mathcal{K}_D$ is of the same order as that of $\mathcal{F}_D$.

Lemma 2. *For any* $\boldsymbol{f}(\boldsymbol{x}; \Theta), \boldsymbol{f}'(\boldsymbol{x}; \Theta') \in \mathcal{K}_D(W, L, B, M)$*, it holds true that*

$$\sup_{\|\boldsymbol{x}\|_\infty \le 1} \|\boldsymbol{f}(\boldsymbol{x}; \Theta) - \boldsymbol{f}'(\boldsymbol{x}; \Theta')\|_2 \le pC(W, L, B)\epsilon,$$

provided that $\|\Theta - \Theta'\|_\infty \le \epsilon$*, where* $C(W, L, B) = (WB)^L(\frac{L}{B} + \frac{L}{WB-1}) - \frac{(WB)^L - 1}{(WB-1)^2}$*.*

Lemma 2 establishes a Hölder-type continuity property for the neural networks in $\mathcal{K}_D(W, L, B, M)$, where $C(W, L, B)$ may diverge to infinity with W, L and B. This continuity property paves the road for computing the entropy of the functional class for the user and item neural networks as in Lemma 3.

Lemma 3. *For any* $\Phi = (W, L, B, M, \tilde{W}, \tilde{L}, \tilde{B})$*, it holds true that*

$$\log \mathcal{N}_{[\cdot]}\big(\epsilon, \mathcal{R}^{\Phi}, \|\cdot\|_{L^2(\mu_{ui})}\big) \le C_2(W \log W + \tilde{W} \log \tilde{W}) \log\Big(\epsilon^{-1} C_3(C(W, L, B) + C(\tilde{W}, \tilde{L}, \tilde{B}))\Big),$$

where $\mathcal{N}_{[\cdot]}(\epsilon, \mathcal{R}^{\Phi}, \|\cdot\|_{L^2(\mu_{ui})})$ *is the* ϵ*-bracketing number of* $\mathcal{R}^{\Phi}$ *under the* $\|\cdot\|_{L^2(\mu_{ui})}$ *metric,* $C_2 = 28 \max\{L, \tilde{L}\}$*,* $C_3 = 2p^{3/2}M \max\{B, \tilde{B}\}$*, and* $C(\cdot, \cdot, \cdot)$ *is defined as in Lemma 2.*

Lemma 3 establishes an upper bound on the bracketing entropy of the two-tower model, which provides the key ingredient for deriving the estimation error of the two-tower model based on the empirical process theory and some large-deviation inequalities. Similar entropy measures have also been used to quantify the richness of various functional classes as in Zhou (2002) and Zhang (2002).

Theorem 2. *Suppose all the assumptions in Theorem 1 are met. Then there holds truth that*

$$P\Big(\|\hat{R} - R^*\|^2_{L^2(\mu_{ui})} \le L_{ui}|\Omega|^{-2\beta/(2\beta+d_{ui})}(\log|\Omega|)^2\Big) \ge 1 - 24\exp(-C_1|\Omega|^{d_{ui}/(2\beta+d_{ui})}\log|\Omega|),$$

provided that $4\lambda_{|\Omega|}J(R_0) \leq L_{ui}|\Omega|^{-2\beta/(2\beta+d_{ui})}\log|\Omega|$, *where* $L_{ui} = \max\{L, \tilde{L}\}$ *with* $L = O(\beta\log_2\beta/d_u)$ *and* $\tilde{L} = O(\beta\log_2\beta/d_i)$, $C_1 = 6\max\{(50p^2M^4+4\sigma^2), 1\}(25p^2M^4+B_e^2)/13$, $B_e = O(|\Omega|^c)$ *with* $c < d_{ui}/(4\beta+2d_{ui})$, $d_{ui} = \max\{d_u, d_i\}$, $W = O(|\Omega|^{d_{ui}/(2\beta+d_{ui})}\log|\Omega|)$, $\tilde{W} = O(|\Omega|^{d_{ui}/(2\beta+d_{ui})}\log|\Omega|)$, $B = O(|\Omega|^{2\beta s/(2\beta+d_u)})$, *and* $\tilde{B} = O(|\Omega|^{2\beta s/(2\beta+d_i)})$.

Theorem 2 shows that the two-tower model converges to the true model at some fast rate, which is explicitly governed by the values of d_u, d_i and β. Particularly, when β is sufficiently large, the convergence rate will be approximately $O_p(|\Omega|^{-1}(\log|\Omega|)^2)$, which is faster than most existing results in literature (Zhu et al., 2016). This advantage is mainly due to the fact that the latent embeddings of user and item provide a smooth representation of covariates, and hence the number of parameters of the two-tower model is significantly less than that of the classical collaborative filtering methods, leading to a faster rate of convergence. Moreover, it is interesting to note that L_{ui} is fixed when β, d_u, and d_i are pre-specified, suggesting that finite depths of the two-tower model are sufficient for approximating the true model with the widths of user network and item network increasing at the rate $O(|\Omega|^{d_{ui}/(2\beta+d_{ui})}\log|\Omega|)$.

4 Numerical experiments

In this section, we examine the numerical performance of the two-tower model, denoted as TTM, in various synthetic and real-life examples, and compare it against a number of existing competitors, including regularized SVD (rSVD; Salakhutdinov et al., 2007), SVD++ (SVD$_{pp}$; Koren, 2008), co-clustering algorithm (Co-Clust; George and Merugu, 2005), and K-nearest neighbors (KNN). Whereas TTM is implemented via TensorFlow, the implementations of all other methods are available in the Python package "surprise" (Hug, 2020). Specifically, rSVD employs an alternative least square (ALS) algorithm to estimate latent factors for users and items iteratively (Koren et al., 2009; Dai et al., 2019); SVD++ employs stochastic gradient descent (SGD) to minimize a regularized square error objective; Co-Clust divides users and items into clusters which are assigned different baseline ratings; SlopeOne is basically an item-based collaborative filtering method in leveraging ratings of other similar items for prediction; KNN mainly utilizes the weighted average of ratings of top-K most similar users for prediction.

The tuning parameters for all methods are determined by grid search. Particularly, we split each data set into two subsets for training and testing. Then, the optimal model of SVD$_{pp}$, KNN, rSVD and Co-Clust will be determined via 5-fold cross-validation based on training data set, and that of TTM is determined by a validation set with ratio to training set being 0.2 for saving computational cost encountered in cross-validation. The grids for the regularization parameter λ in TTM and rSVD are set as $\{10^{-6+k/3}; k = 0, \ldots, 24\}$, and the grids for the number of clusters in Co-Clust and neighborhood parameter K in KNN are set as $\{5, 10, 15, \ldots, 50\}$. The similarity measure in KNN is set as mean square similarity difference between common ratings for any pair of users or items (Hug, 2020). Furthermore, as TTM replies on deep neural network, the learning rate of SGD is set as 0.01 in the beginning with decay rate and minimal learning rate being 0.9 and 0.005, respectively, and an early-stopping scheme is employed to avoid over-fitting.

4.1 Synthetic examples

We consider various scenarios of a synthetic example. First, the sizes of the rating matrix $\boldsymbol{K} = \{k_{ui}\}_{1\leq u\leq n; 1\leq i\leq m}$ are set as $(n, m) = (1500, 1500), (2000, 2000)$ and $(3000, 3000)$, and the number of observed ratings is fixed at 100,000, which amounts to sparsity ranging from 0.011 to 0.044. Second, we set the nominal dimensions of $\boldsymbol{x}_u$ and $\tilde{\boldsymbol{x}}_i$ as $D_u = D_i = 50$, representation dimension as $p = 30$, and the true functions for users and items as $\boldsymbol{f}^*(\boldsymbol{x}_u) = (f_1^*(\boldsymbol{x}_u), \ldots, f_p^*(\boldsymbol{x}_u))$ and $\tilde{\boldsymbol{f}}^*(\tilde{\boldsymbol{x}}_i) = (\tilde{f}_1^*(\tilde{\boldsymbol{x}}_i), \ldots, \tilde{f}_p^*(\tilde{\boldsymbol{x}}_i))$, where $f_j^*(\boldsymbol{x}_u) = \sum_{l=1}^{D_u}\alpha_{jl}\sin(2\pi x_{ul}) + \sum_{l=1}^{D_u}\beta_{jl}\cos(2\pi x_{ul}) + \sum_{l=1}^{D_u-1}\zeta_{jl}x_{ul}x_{u(l+1)}$ and $\tilde{f}_j^*(\tilde{\boldsymbol{x}}_i) = \sum_{l=1}^{D_i}\tilde{\alpha}_{jl}\sin(2\pi\tilde{x}_{il}) + \sum_{l=1}^{D_i}\tilde{\beta}_{jl}\cos(2\pi\tilde{x}_{ul}) + \sum_{l=1}^{D_i-1}\tilde{\zeta}_{jl}\tilde{x}_{ul}\tilde{x}_{u(l+1)}$ with $\alpha_{jl}, \tilde{\alpha}_{jl}, \beta_{jl}, \tilde{\beta}_{jl}, \zeta_{jl}$ and $\tilde{\zeta}_{jl}$ being uniformly sampled from $[-0.15, 0.15]$. To mimic the low intrinsic dimension of covariates, we sample x_{ul} and $\tilde{x}_{il}$ from $[0, 1]$ for $l = 1, \ldots, d$, and set $x_{ul} = x_{u(l-d)}$ and $\tilde{x}_{il} = \tilde{x}_{i(l-d)}$ for $l = d+1, \ldots, 50$, where the intrinsic dimension $d \in \{20, 30, 40\}$. Finally, the ratings are generated from the following model,

$$k_{ui} = \langle \boldsymbol{f}^*(\boldsymbol{x}_u), \tilde{\boldsymbol{f}}^*(\tilde{\boldsymbol{x}}_i)\rangle + \epsilon_{ui},$$

where ϵ_{ui} is set as a Gaussian distribution with mean 0 and variance 0.1.

In each scenario, the neural networks for both user and item in TTM are set as 5-layer fully-connected neural network with 50 neurons in each hidden layers and 30 neurons in the output layer. The averaged root mean square errors (RMSEs) of each method as well as their standard errors are summarized in Table 1. Table 1 shows that TTM yields

Table 1: Averaged RMSE of various methods as well as their standard errors (in parentheses) over 50 replications. The best performers in each case are bold-faced.

(n, m, d)	TTM	rSVD	KNN	SVD$_{pp}$	Co-Clust
(1500,1500,20)	**0.496(0.011)**	1.566(0.008)	1.990(0.013)	1.507(0.008)	1.815(0.012)
(1500,1500,30)	**1.330(0.010)**	1.742(0.009)	2.063(0.011)	1.704(0.007)	1.944(0.010)
(1500,1500,40)	**1.604(0.011)**	1.845(0.007)	2.075(0.012)	1.806(0.007)	2.015(0.011)
(2000,2000,20)	**0.438(0.022)**	1.908(0.010)	2.074(0.016)	1.849(0.008)	1.907(0.013)
(2000,2000,30)	**1.358(0.013)**	2.041(0.013)	2.120(0.012)	1.995(0.011)	2.027(0.013)
(2000,2000,40)	**1.703(0.009)**	2.110(0.010)	2.150(0.010)	2.073(0.009)	2.089(0.010)
(3000,3000,20)	**0.373(0.010)**	2.105(0.023)	2.301(0.024)	2.198(0.021)	2.149(0.022)
(3000,3000,30)	**1.353(0.013)**	2.196(0.012)	2.311(0.012)	2.204(0.012)	2.246(0.015)
(3000,3000,40)	**1.862(0.010)**	2.209(0.020)	2.338(0.021)	2.219(0.019)	2.291(0.021)

smallest test errors in all cases with improvement ranging from 12.6% to 81.3%. Particularly, the improvement becomes more substantial when n and m increase and the rating matrix becomes sparser. This is expected as the existing method suffers from the severe cold-start issue in sparse rating matrix. By sharp contrast, TTM is much more robust to sparse rating matrix when the intrinsic dimensionality of covariates is small, and hence that it can greatly circumvent the cold-start issue. This provides numerical support for the established theoretical results in Theorem 2, which shows that the convergence rate of TTM increases as d decreases.

4.2 YELP DATASET

In this section, we apply the two-tower model to the Yelp challenge dataset, which is publicly available at `https://www.yelp.com/dataset`. The data set has four interrelated parts relating to "user", "business", "review", and "check-in", respectively. In "user" part, personal information about almost 5,200,000 users in Yelp community is available, including number of reviews, fans count, elite experience, and personal social network. Additionally, users' behavior like averaged stars of reviews and voting obtained from other users like "useful", "funny" and "cool" are also given. In "business" part, the location, latitude-longitude, review counts, and categories of almost 174,000 businesses are given. In "review", each review consists of user, business, textual comment, and corresponding stars for the business. In "check-in" part, the counts of check-ins at each business are provided. We construct profiles of users and businesses by using parts "user", "business", and "review" for constructing profiles for users and businesses, which will be used as covariates in the TTM.

To process the data set, we focus cities with at least 20 businesses and businesses with at least 100 reviews, which amounts to 15,090 businesses in the item set. For each business, we numericalize "location" and "category" by one-hot encoding and use them as part of item covariates. For users, we collect their binarized elite experience indicating whether they have ever acted as elite user in Yelp community and overall feedbacks they obtained including "useful", "cool", and "funny". Furthermore, we construct covariates for users and businesses based on textual reviews. Specifically, we collect all textual reviews and employ term frequency-inverse document frequency (tf-idf) to extract 300 most significant 1-gram, 2-gram, and 3-gram. In this manner, each review is converted into an integer covariate vector of length 300 based on bag-of-words technique. For a specific user or business, we averaged all the bag-of-words representations of its reviews, and then concatenate it with covariates constructed in the earlier step.

Additionally, we notice an interesting phenomena that users usually comment on aspects of restaurants using words with polarity such as "Oh yeah! Not only that the service was good, the food is good the serving is good and the service is amazing", "Jamie our waitress is so sweet and attentive.", and so on. In the first example, the user uses "good" and "amazing" to describe "food" and "service" in this restaurant, and in the second example the user uses "sweet" and "attentive" to describe "waitress". Intuitively, comments on aspects characterize features of restaurants, and also aspects appearing frequently in a user's reviews also indicate what he/she cares most in the process of consumption. To capture such information, we utilize the Python package "Spacy" to capture entities and corresponding sentiment words given by the function "SentimentIntensityAnalyzer" in the Python package "nltk". As shown in Table 2, users are apt to provide feedbacks over "food", "service", "place" and "staff" in reviews as expected. Additionally, it is interesting to note that "price" obtains a relatively lower averaged polarity score in reviews, which is only 0.28 in stark contrast with other aspects. This implies that reviews containing "price" are more likely to have lower ratings. Finally, we select 200 most common aspects in reviews and construct vectors of length 200 with elements being averaged polarity scores of associated aspects in reviews of a user or business.

Table 2: Descriptive statistics of polarity scores of 10 most common aspects of in selected reviews.

aspect	frequency	mean	std	25%	50%	75%
atmosphere	7096	0.42	0.21	0.40	0.44	0.56
food	66879	0.39	0.27	0.36	0.44	0.57
fries	7772	0.39	0.28	0.42	0.44	0.57
place	32201	0.34	0.32	0.32	0.43	0.57
prices	7179	0.28	0.32	0.23	0.43	0.44
salad	6508	0.38	0.27	0.32	0.44	0.57
sauce	7081	0.41	0.28	0.44	0.46	0.57
server	11874	0.43	0.24	0.42	0.49	0.56
service	71755	0.40	0.30	0.44	0.49	0.57
staff	29270	0.45	0.19	0.42	0.49	0.49

After the pre-processing step, we obtain a data set containing 15,090 business, 35,906 users and 688,960 ratings. We replicate the numerical experiments 50 times, and in each replication, we randomly choose 15,000 users and 10,000 businesses, as well as their observed ratings for experiment. We split the selected data set in training and testing sets with the ratio 70-30, and follow the tuning process as described in the beginning of Section 4. Moreover, the remaining reviews will be used for evaluating the performance of the TTM in the cold-start setting.

Table 3: Averaged RMSE of various methods as well as their standard errors (in parentheses) over 50 replications. The best performers in each case are bold-faced.

	TTM	rSVD	KNN	SVD$_{pp}$	Co-Clust
Overall	**0.9624** (0.0004)	1.0325 (0.004)	1.0544 (0.0004)	1.0338 (0.0003)	1.0573 (0.0004)
Warm-Start	**0.9547** (0.0007)	0.9655 (0.0007)	1.0449 (0.0008)	0.9703 (0.0006)	1.0553 (0.0008)
Cold-Start	**0.9654** (0.0004)	1.0581 (0.0002)	1.0581 (0.0002)	1.0581 (0.0002)	1.0581 (0.0002)

Table 3 shows that TTM yields the smallest test errors in terms of overall recommendations, with improvement ranging from 6.79% to 8.98%. These improvements are mostly due to the improved recommendation accuracy on the cold-start pairs, for which all other methods yield the same RMSE as 1.0581. This clearly demonstrates that TTM is capable of leveraging high-dimensional user and item covariates to capture underlying interaction between users and items, and thus enable accurate recommendations for cold-start pairs without any observed ratings.

5 Summary

This paper quantifies the asymptotic convergence of the two-tower model to the optimal recommender system, which integrates multiple covariate sources of information to improve recommendation accuracy. The two-tower model consists of two deep neural networks to embed users and items in a low-dimensional numerical space, and estimates ratings through the well-established collaborative filtering structure. It takes advantages of the learning capability of deep neural network to extract informative representations of covariates in an non-linear fashion. Most importantly, this paper provides statistical guarantee of the two-tower model by quantifying its asymptotic behaviors in terms of both approximation error and estimation error. To the best of our knowledge, our established results are among very few theoretical guarantees about the deep recommender systems.

REFERENCES

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Bi, X., Qu, A., Wang, J., and Shen, X. (2017). A group-specific recommender system. *Journal of the American Statistical Association*, 112(519):1344–1353.

Boratto, L., Carta, S., Fenu, G., and Saia, R. (2017). Semantics-aware content-based recommender systems: Design and architecture guidelines. *Neurocomputing*, 254:79–85.

Chen, B., He, S., Li, Z., and Zhang, S. (2012). Maximum block improvement and polynomial optimization. *SIAM Journal on Optimization*, 22(1):87–107.

Dai, B., Wang, J., Shen, X., and Qu, A. (2019). Smooth neighborhood recommender systems. *The Journal of Machine Learning Research*, 20(1):589–612.

Falconer, K. (2004). *Fractal geometry: mathematical foundations and applications*. John Wiley & Sons.

Fortuna, B., Fortuna, C., and Mladenić, D. (2010). Real-time news recommender system. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 583–586. Springer.

George, T. and Merugu, S. (2005). A scalable collaborative filtering framework based on co-clustering. In *Fifth IEEE International Conference on Data Mining (ICDM'05)*, pages 4–pp. IEEE.

Gunawardana, A. and Meek, C. (2009). A unified approach to building hybrid recommender systems. In *Proceedings of the third ACM Conference on Recommender Systems*, pages 117–124.

Hofmann, T. (2004). Latent semantic models for collaborative filtering. *ACM Transactions on Information Systems (TOIS)*, 22(1):89–115.

Hofmann, T. and Puzicha, J. (1999). Latent class models for collaborative filtering. In *IJCAI*, volume 99.

Hug, N. (2020). Surprise: A python library for recommender systems. *Journal of Open Source Software*, 5(52):2174.

Koren, Y. (2008). Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 426–434.

Koren, Y. (2009). Collaborative filtering with temporal dynamics. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 447–456.

Koren, Y., Bell, R., and Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37.

Lang, K. (1995). Newsweeder: Learning to filter netnews. In *Machine Learning Proceedings 1995*, pages 331–339. Elsevier.

Li, Y., Zhang, D., Lan, Z., and Tan, K.-L. (2016). Context-aware advertisement recommendation for high-speed social news feeding. In *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*, pages 505–516. IEEE.

Mao, X., Chen, S. X., and Wong, R. K. (2019). Matrix completion with covariate information. *Journal of the American Statistical Association*, 114(525):198–210.

Mazumder, R., Hastie, T., and Tibshirani, R. (2010). Spectral regularization algorithms for learning large incomplete matrices. *The Journal of Machine Learning Research*, 11:2287–2322.

Middleton, S. E., Shadbolt, N. R., and De Roure, D. C. (2004). Ontological user profiling in recommender systems. *ACM Transactions on Information Systems (TOIS)*, 22(1):54–88.

Miller, B. N., Albert, I., Lam, S. K., Konstan, J. A., and Riedl, J. (2003). Movielens unplugged: experiences with an occasionally connected recommender system. In *Proceedings of the 8th International Conference on Intelligent User Interfaces*, pages 263–266.

Nakada, R. and Imaizumi, M. (2020). Adaptive approximation and generalization of deep neural network with intrinsic dimensionality. *Journal of Machine Learning Research*, 21(174):1–38.

Oku, K., Nakajima, S., Miyazaki, J., and Uemura, S. (2006). Context-aware svm for context-dependent information recommendation. In *7th International Conference on Mobile Data Management (MDM'06)*, pages 109–109. IEEE.

Romadhony, A., Al Faraby, S., and Pudjoatmodjo, B. (2013). Online shopping recommender system using hybrid method. In *2013 International Conference of Information and Communication Technology (ICoICT)*, pages 166–169. IEEE.

Salakhutdinov, R., Mnih, A., and Hinton, G. (2007). c. In *Proceedings of the 24th international conference on Machine learning*, pages 791–798.

Subramaniyaswamy, V. and Logesh, R. (2017). Adaptive knn based recommender system through mining of user preferences. *Wireless Personal Communications*, 97(2):2229–2247.

Twardowski, B. (2016). Modelling contextual information in session-aware recommender systems with neural networks. In *Proceedings of the 10th ACM Conference on Recommender Systems*, pages 273–276.

Van den Oord, A., Dieleman, S., and Schrauwen, B. (2013). Deep content-based music recommendation. In *Advances in Neural Information Processing Systems*, pages 2643–2651.

Vargas-Govea, B., González-Serna, G., and Ponce-Medellın, R. (2011). Effects of relevant contextual features in the performance of a restaurant recommender system. *ACM RecSys*, 11(592):56.

Wang, J., Shen, X., Sun, Y., and Qu, A. (2016). Classification with unstructured predictors and an application to sentiment analysis. *Journal of the American Statistical Association*, 111(515):1242–1253.

Wang, J., Yessenalina, A., and Roshan-Ghias, A. (2021a). Exploring heterogeneous metadata for video recommendation with two-tower model. *arXiv preprint arXiv:2109.11059*.

Wang, J., Zhu, J., and He, X. (2021b). Cross-batch negative sampling for training two-tower recommenders. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1632–1636.

Xu, S., Dai, B., and Wang, J. (2021). Sentiment analysis with covariate-assisted word embeddings. *Electronic Journal of Statistics*, 15(1):3015–3039.

Yang, D., Zhang, J., Wang, S., and Zhang, X. (2019). A time-aware cnn-based personalized recommender system. *Complexity*, 2019.

Yang, J., Yi, X., Zhiyuan Cheng, D., Hong, L., Li, Y., Xiaoming Wang, S., Xu, T., and Chi, E. H. (2020). Mixed negative sampling for learning two-tower neural networks in recommendations. In *Companion Proceedings of the Web Conference 2020*, pages 441–447.

Yi, X., Yang, J., Hong, L., Cheng, D. Z., Heldt, L., Kumthekar, A., Zhao, Z., Wei, L., and Chi, E. (2019). Sampling-bias-corrected neural modeling for large corpus item recommendations. In *Proceedings of the 13th ACM Conference on Recommender Systems*, pages 269–277.

Yu, T., Shen, Y., and Jin, H. (2019). A visual dialog augmented interactive recommender system. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 157–165.

Zhang, T. (2002). Covering number bounds of certain regularized linear function classes. *Journal of Machine Learning Research*, 2(Mar):527–550.

Zhou, D.-X. (2002). The covering number in learning theory. *Journal of Complexity*, 18(3):739–767.

Zhu, Y., Shen, X., and Ye, C. (2016). Personalized prediction and sparsity pursuit in latent factor models. *Journal of the American Statistical Association*, 111(513):241–252.