A General Cross-domain Recommendation Framework via Bayesian Neural Network

Jia He[1,2], Rui Liu[3], Fuzhen Zhuang[1,2,*], Fen Lin[4], Cheng Niu[4], Qing He[1,2]
[1]Key Lab of Intelligent Information Processing of Chinese Academy of Sciences (CAS),
Institute of Computing Technology, CAS, Beijing 100190, China
[2]University of Chinese Academy of Sciences, Beijing 100049, China, {hejia, zhuangfuzhen, heqing}@ict.ac.cn
[3]School of Computer Science and Technology, Dalian University of Technology, Dalian, China, 913917030@mail.dlut.edu.cn
[4]WeChat Search Application Department, Tencent, Beijing, China, {felicialin, niucheng}@tencent.com
* The corresponding author

***Abstract*—Collaborative filtering is an effective and widely used recommendation approach by applying the user-item rating matrix for recommendations, however, which usually suffers from cold-start and sparsity problems. To address these problems, hybrid methods are proposed to incorporate auxiliary information such as user/item profiles to collaborative filtering models; Cross-domain recommendation systems add a new dimension to solve these problems by leveraging ratings from other domains to improve recommendation performance. Among these methods, deep neural network based recommendation systems achieve excellent performance due to their excellent ability in learning powerful representations. However, these cross-domain recommendation systems based on deep neural network rarely consider the uncertainty of weights. Therefore, they maybe lack of calibrated probabilistic predictions and make overly confident decisions. Along this line, we propose a general cross-domain recommendation framework via Bayesian neural network to incorporate auxiliary information, which takes advantage of both the hybrid recommendation methods and the cross-domain recommendation systems. Specifically, our framework consists of two kinds of neural networks, one to learn the low dimensional representation from the one-hot codings of users/items, while the other one is to project the auxiliary information of users/items into another latent space. The final rating is produced by integrating the latent representations of the one-hot codings of users/items and the auxiliary information of users/items. The latent representations of users learnt from ratings and auxiliary information are shared across different domains for knowledge transfer. Moreover, we capture the uncertainty in all weights by representing weights with Gaussian distributions to make calibrated probabilistic predictions. We have done extensive experiments on real-world data sets to verify the effectiveness of our framework.**

***Keywords*-Cross-domain learning, Recommendation systems, Bayesian neural network**

I. Introduction

With the explosively growing amount of online information, recommendation system (RS) plays an essential role in alleviating information overload problem and is widely used by many websites, e.g., Amazon, YouTube and Netflix. In recent years, RS has attracted a vast amount of interest, and a great deal of research has been proposed to improve the recommendation performance [1], [2]. Collaborative filtering (CF) is a popular and widely used recommendation approach in which the preference of a user on an item is predicted based on the preferences of other users with similar interest [3]–[6]. However, the recommendation quality of CF heavily depends on the user-item rating matrix, which is often highly sparse in real-world situations, leading to the degradation of recommendation performance. Moreover, CF based on the sole rating matrix is not applicable for newly joined users or items, which is known as the cold-start problem.

To address these problems, there are two pipelines of recommender systems, i.e., hybrid recommendations and cross-domain recommendations. Hybrid recommendation methods alleviate the data sparsity and cold-start problems and enhance recommendation performance by incorporating the auxiliary information such as items' profiles or users' profiles/social networks [7]–[10] as regularization terms. Deep learning techniques recently show great potential for learning effective representations from features and deliver state-of-the-art performance in many application [11], [12]. So recently some hybrid recommendation systems adopt deep neural networks to learn latent representation from auxiliary information and have achieved excellent performance in single-domain recommendation tasks [5], [13]–[16].

On the other side, cross-domain recommendation systems (CDRS) are also proposed to address the sparsity problem by leveraging the rating information from other domains to enhance the prediction on the target domain [7], [17]. Existing CDRS can be roughly classified into two major categories. In the first category, patterns learned from the source domain are directly transferred to the target domain to improve recommendation accuracy in the target domain [18]–[20]. For example, codebook transfer (CBT) [18] which transfers user-item rating patterns from a dense rating matrix in a source domain to a sparse rating matrix in a target domain. And rating-matrix generative model (RMGM) [19] considers the cluster-level rating patterns as potential candidates to be transferred from the source domain. These methods usually focus on transferring knowledge from a single source domain to a target domain. The second category of methods learn the models from the source and target domains simultaneously, and expect that these domains can complement each other [2], [7], [17], [21], [22]. In this case, there is no distinction between the source domain and the target domain, i.e., both domains are treated equally. All domains work in a collective way. One of the representative work is EMCDR [2] which captures the nonlinear mapping function between source domain and target domain with a multi-layer perceptron. However, most CDRS only transfer patterns across domains and don't incorporate the extra auxiliary information from users or items. Therefore, when the newly joined user or item does not exist in any domain, it is difficult for these traditional CDRS to make recommendation. To address this cold-start problem, MVDNN [23] uses a deep learning approach to map users'/items' auxiliary information to a latent space and jointly learns the latent factors of users/items across different domains. CCCFNet [24] is a cross-domain recommendation system based on neural network, which introduces items' auxiliary information to collaborative filtering with neural network. But it does not incorporate the auxiliary information of users, thus it is difficult for CCCFNet to recommend items to newly joined users.

Although some state-of-the-art CDRS like MVDNN and CCCFNet have achieved good performance by using deep learning to learn powerful representations from rich auxiliary information, they obtain point estimates of the weights by optimizing a objective function, which may lead to make overly confident predictions without considering the uncertainty of weights and may be prone to over-fitting the data when the network is too complex. To overcome these problems, the Bayesian deep learning is proposed [25]. But exact Bayesian inference on the weights of a neural network is intractable as the number of parameters is very large and the function form of a neural network does not lend itself to exact integration. Bayes by Backprop (BBB) [26] was proposed to take a variational approximation to exact Bayesian updates.

To this end, we propose a **G**eneral **C**ross-domain framework via **BA**yesian **N**eural network (namely GCBAN) to incorporate the auxiliary information from both users and items, which can take advantage of both hybrid recommendation approach and cross-domain approach. Specifically, our framework consists of two kinds of neural networks, one to learn the low dimensional representation from the one-hot codings of users/items, while the other one is to project the auxiliary information of users/items into another latent space. The final rating is produced by integrating the latent representations of the one-hot codings of users/items and the auxiliary information of users/items. The latent representations of users learnt from ratings and auxiliary information are shared across different domains for knowledge transfer. Furthermore, we capture the uncertainty of all weights which are represented by Gaussian distributions with BBB. Finally, We have done extensive experiments on real-world data sets to verify the effectiveness of our framework. Note that, the user set are shared by all domains on our experimental datasets.

II. Preliminary Knowledge

A. Notations and Problem Description

The frequently used notations and denotations are shown in Table I. Actually, our model is a general framework, which

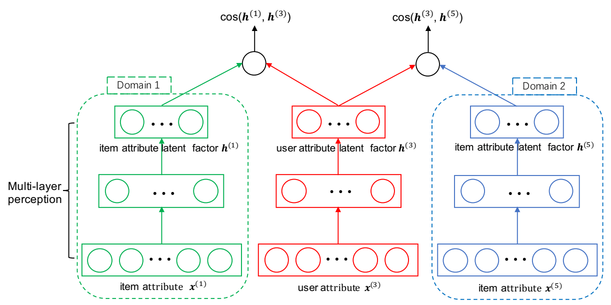
TABLE I: Notations and Denotations

Symbol	Description
$\boldsymbol{X}^{(1)}$, $\boldsymbol{X}^{(2)}$	The one-hot encoding and attribute matrix of items in $\mathcal{D}_1$
$\boldsymbol{X}^{(3)}$, $\boldsymbol{X}^{(4)}$	The one-hot encoding and attribute matrix of users
$\boldsymbol{X}^{(5)}$, $\boldsymbol{X}^{(6)}$	The one-hot encoding and attribute matrix of items in $\mathcal{D}_2$
$\boldsymbol{r}^{(1)}$, $\boldsymbol{r}^{(2)}$	The real rating vector in $\mathcal{D}_1, \mathcal{D}_2$
$\boldsymbol{W}^{(i,j)}$, $\boldsymbol{b}^{(i,j)}$	The weight matrix and bias vector
$\boldsymbol{h}^{(1)}$, $\boldsymbol{h}^{(2)}$	The attribute latent vector and latent factor of items in $\mathcal{D}_1$
$\boldsymbol{h}^{(3)}$, $\boldsymbol{h}^{(4)}$	The attribute latent vector and latent factor of users
$\boldsymbol{h}^{(5)}$, $\boldsymbol{h}^{(6)}$	The attribute latent vector and latent factor of items in $\mathcal{D}_2$
$\boldsymbol{y}^{(1)}$, $\boldsymbol{y}^{(2)}$	The learned rating vector in $\mathcal{D}_1, \mathcal{D}_2$
μ, ρ	The parameters of variational posterior distributions
δ_0	The standard deviation of the likelihood distribution
γ_0	The standard deviation of the prior distribution
$\odot$	The element-wise multiplication
$\cup$	Union of two sets

can deal with multiple domains for recommendations (e.g., any number of source domains and any number of target domains). For concision, in this paper we only focus on two domain recommendations, i.e., one for source domain and the other one for target domain. Given two domains (domain 1 denoted as $\mathcal{D}_1$ and domain 2 denoted as $\mathcal{D}_2$). There are $T^{(1)}$ ratings in $\mathcal{D}_1$, $T^{(2)}$ ratings in $\mathcal{D}_2$, $N^{(o)}$ users (two domains share the same set of users), $M^{(o,1)}$ items in $\mathcal{D}_1$, $M^{(o,2)}$ items in $\mathcal{D}_2$, $N^{(a)}$ attributes of users, $M^{(a,1)}$ attributes of items in $\mathcal{D}_1$ and $M^{(a,2)}$ attributes of items in $\mathcal{D}_2$. Each rating in $\mathcal{D}_1/\mathcal{D}_2$ consists of five parts: the attribute vector and the one-hot encoding vector of the user, the attribute vector and the one-hot encoding vector of the item and the rating value, denoted as $\{\boldsymbol{x}_t^{(3)}, \boldsymbol{x}_t^{(4)}, \boldsymbol{x}_t^{(1)}, \boldsymbol{x}_t^{(2)}, r_t^{(1)}\}$, $t \in \{1, \cdots, T^{(1)}\}$. Similarly, each rating in $\mathcal{D}_2$ is denoted as $\{\boldsymbol{x}_{T^{(1)}+t}^{(3)}, \boldsymbol{x}_{T^{(1)}+t}^{(4)}, \boldsymbol{x}_t^{(5)}, \boldsymbol{x}_t^{(6)}, r_t^{(2)}\}$, $t \in \{1, \cdots, T^{(2)}\}$. Our goal is to propose a new cross-domain framework to make full use of all the data for making accurate recommendations.

B. MVDNN

MVDNN [23] is a multi-view deep learning approach for cross-domain in recommendation systems. It uses a deep learning approach to map users and items to a latent space where the similarity between users and their preferred items is maximized, and the items' features are jointly learned from different domains. In this work, we propose our cross-domain framework motivated by MVDNN, and next we first briefly review the MVDNN model.


$\cos(\boldsymbol{h}^{(1)}, \boldsymbol{h}^{(3)})$
$\cos(\boldsymbol{h}^{(3)}, \boldsymbol{h}^{(5)})$
Domain 1
Domain 2
item attribute latent factor $\boldsymbol{h}^{(1)}$
user attribute latent factor $\boldsymbol{h}^{(3)}$
item attribute latent factor $\boldsymbol{h}^{(5)}$
Multi-layer perception
item attribute $\boldsymbol{x}^{(1)}$
user attribute $\boldsymbol{x}^{(3)}$
item attribute $\boldsymbol{x}^{(5)}$
Fig. 1: MVDNN

In MVDNN, users' features can be seen as a view and items' attributions are regarded as another view. The architecture of the MVDNN for two domains is shown in Fig. 1, which can be easily extended to multiple domains. The pivot view is the user view denoted as $\boldsymbol{X}^{(3)}$ and other two auxiliary views from $\mathcal{D}_1$ and $\mathcal{D}_2$ are represented as $\boldsymbol{X}^{(1)}$ and $\boldsymbol{X}^{(5)}$ respectively. MVDNN uses several non-linear mapping layers to project the input $\boldsymbol{x}^{(1)}$ /$\boldsymbol{x}^{(3)}$/$\boldsymbol{x}^{(5)}$ into shared semantic space $\boldsymbol{h}^{(1)}$/$\boldsymbol{h}^{(3)}$/$\boldsymbol{h}^{(5)}$. Let $k^{(3)}$, $k^{(1)}$and $k^{(5)}$ represent the number of layers in the neural network of users, items in $\mathcal{D}_1$ and items in $\mathcal{D}_2$, $\boldsymbol{l}^{(i,j)}$ $(i \in \{1,3,5\}, j = 1, \cdots, k^{(i)}-1)$ denotes the j-th hidden layer, $\boldsymbol{W}^{(i,j)}$ and $\boldsymbol{b}^{(i,j)}$ represent the weight matrix and bias term of the j-th layer in i-th network, $\boldsymbol{h}^{(i)}$ denote the final output latent representations:

$$\begin{aligned}\boldsymbol{l}^{(i,1)} &= f(\boldsymbol{x}^{(i)}\boldsymbol{W}^{(i,1)} + \boldsymbol{b}^{(i,1)}),\\ \boldsymbol{l}^{(i,j)} &= f(\boldsymbol{l}^{(i,j-1)}\boldsymbol{W}^{(i,j)} + \boldsymbol{b}^{(i,j)}),\\ \boldsymbol{h}^{(i)} &= f(\boldsymbol{l}^{(i,k^{(i)}-1)}\boldsymbol{W}^{(i,k^{(i)})} + \boldsymbol{b}^{(i,k^{(i)})}),\\ i \in& \{1,3,5\}, \quad j = 2, \cdots, k^{(i)}-1,\end{aligned} \tag{1}$$

where $f(\cdot)$ represents the non-linear activation function. Then we can obtain the learnt rating $y^{(1)}/y^{(2)}$ from $\mathcal{D}_1/\mathcal{D}_2$:

$$y^{(1)} = e^{\alpha_1 \cos(\boldsymbol{h}^{(3)},\boldsymbol{h}^{(1)})}, \quad y^{(2)} = e^{\alpha_2 \cos(\boldsymbol{h}^{(3)},\boldsymbol{h}^{(5)})}. \tag{2}$$

The real rating value is $r^{(1)}$ / $r^{(2)}$ and the objective function is to minimize the reconstruction error between the real ratings and learned ratings. The objective function can be formulated as follows:

$$L = \sum_{t}^{T^{(1)}} (y_t^{(1)} - r_t^{(1)})^2 + \sum_{t}^{T^{(2)}} (y_t^{(2)} - r_t^{(2)})^2. \tag{3}$$

III. A General Cross-domain Recommendation Framework via Bayesian Neural Network

As introduced above, MVDNN only uses the auxiliary information and does not take the collaborative filtering into account. In this way, it doesn't take advantage of relevance between users and items. What's more, it doesn't consider the uncertainty of weights in the network and only can obtain point estimates of the weights, which may lead to the lack of calibrated probabilistic predictions and make overly confident predictions. So we propose a new recommendation framework GCBAN to incorporate the auxiliary information from both users and items.

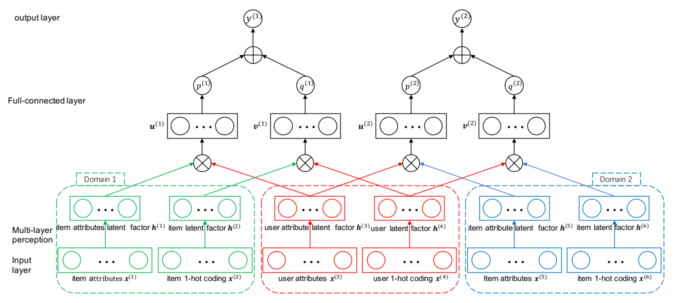

output layer
$y^{(1)}$
$y^{(2)}$
Full-connected layer
$p^{(1)}$
$q^{(1)}$
$p^{(2)}$
$q^{(2)}$
$\boldsymbol{u}^{(1)}$
$\boldsymbol{v}^{(1)}$
$\boldsymbol{u}^{(2)}$
$\boldsymbol{v}^{(2)}$
Domain 1
Domain 2
Multi-layer perception
item attributes latent factor $\boldsymbol{h}^{(1)}$
item latent factor $\boldsymbol{h}^{(2)}$
user attribute latent factor $\boldsymbol{h}^{(3)}$
user latent factor $\boldsymbol{h}^{(4)}$
item attribute latent factor $\boldsymbol{h}^{(5)}$
item latent factor $\boldsymbol{h}^{(6)}$
Input layer
item attributes $\boldsymbol{x}^{(1)}$
item 1-hot coding $\boldsymbol{x}^{(2)}$
user attributes $\boldsymbol{x}^{(3)}$
user 1-hot coding $\boldsymbol{x}^{(4)}$
Item attributes $\boldsymbol{x}^{(5)}$
item 1-hot coding $\boldsymbol{x}^{(6)}$

Fig. 2: GCBAN

The architecture of our model is illustrated in Fig. 2. For $\mathcal{D}_1$, a multi-layer perceptron (MLP) is adopted to learn a common low dimensional space for the one-hot encodings of users and items. Then these two kind of latent representations are concatenated into a new vector $\boldsymbol{v}^{(1)}$ by element-wise multiplication. Next, the new vector $\boldsymbol{v}^{(1)}$ is fed into a one-layer perception to produce a value $q^{(1)}$. Similarly, the attributes of users and items are mapped into another shared latent space with another MLP. To incorporate the auxiliary information from users and items, these two kinds of latent factors of attributions are concatenated into a new vector $\boldsymbol{u}^{(1)}$ by element-wise multiplication. Next, the new attribute vector $\boldsymbol{u}^{(1)}$ is fed into a one-layer perceptron to produce a value $p^{(1)}$. Finally, $p^{(1)}$ and $q^{(1)}$ are combined to obtain the final rating $y^{(1)}$. The recommendation process in $\mathcal{D}_2$ is similar to $\mathcal{D}_1$. These domains share the same user network so that they can complement each other. Specially, we take the uncertainty of all weights and biases in whole network into account to make calibrated probabilistic predictions and avoid over-fitting.

To capture the uncertainty of all weights and biases, we suppose all weights and biases obey some probability distribution. Firstly, we impose Gaussian prior for all weights and biases as follows:

$$\begin{aligned}p(\boldsymbol{W}^{(i,j)}|\gamma_0) &\sim \prod_m \prod_d \mathcal{N}(W_{m,d}^{(i,j)}|0, \gamma_0^2),\\ p(\boldsymbol{b}^{(i,j)}|\gamma_0) &\sim \prod_d \mathcal{N}(b_d^{(i,j)}|0, \gamma_0^2),\\ i \in \{1, \cdots, 6\},& \quad j = 2, \cdots, k^{(i)}-1,\end{aligned} \tag{4}$$

where $\mathcal{N}(\cdot)$ denotes the Gaussian distribution, the mean is 0.

Similarly to Eq.(1), we can obtain the latent factors $\boldsymbol{h}^{(i)}, i \in \{1, \cdots, 6\}$ in the same way. In GCBAN, we use $f = \max(0, x)$ as the activation function $ReLu$ in each layer. The latent representations $\boldsymbol{h}^{(1)}$ and $\boldsymbol{h}^{(3)}$ are projected into a new vector $\boldsymbol{u}^{(1)}$ by element-wise multiplication and the new vector $\boldsymbol{u}^{(1)}$ is mapped into a value $p^{(1)}$ by a one-layer perceptron, and we can get $\boldsymbol{v}^{(1)}$ and $q^{(1)}$ in a similarly way: v

$$\begin{aligned}\boldsymbol{u}^{(1)} &= \boldsymbol{h}^{(1)} \odot \boldsymbol{h}^{(3)}, \quad p^{(1)} = f(\boldsymbol{u}^{(1)}\boldsymbol{W}^{p^{(1)}} + \boldsymbol{b}^{p^{(1)}}),\\ \boldsymbol{v}^{(1)} &= \boldsymbol{h}^{(2)} \odot \boldsymbol{h}^{(4)}, \quad q^{(1)} = f(\boldsymbol{v}^{(1)}\boldsymbol{W}^{q^{(1)}} + \boldsymbol{b}^{q^{(1)}}),\end{aligned} \tag{5}$$

In the same way, we can obtain $p^{(2)}$ and $p^{(2)}$ in $\mathcal{D}_2$. Also, we impose Gaussian prior for all these weights and biases as follows:

$$\begin{aligned}p(\boldsymbol{W}^{p^{(z)}}|\gamma_0) &\sim \prod_m \prod_d \mathcal{N}(W_{m,d}^{p^{(z)}}|0, \gamma_0^2),\\ p(\boldsymbol{b}^{p^{(z)}}|\gamma_0) &\sim \prod_d \mathcal{N}(b_d^{p^{(z)}}|0, \gamma_0^2),\\ p(\boldsymbol{W}^{q^{(z)}}|\gamma_0) &\sim \prod_m \prod_d \mathcal{N}(W_{m,d}^{q^{(z)}}|0, \gamma_0^2),\\ p(\boldsymbol{b}^{q^{(z)}}|\gamma_0) &\sim \prod_d \mathcal{N}(b_d^{q^{(z)}}|0, \gamma_0^2), \quad z \in \{1, 2\}.\end{aligned} \tag{6}$$

Next, we combine $p^{(1)}$ and $q^{(1)}$ to produce the final predicted rating value $y^{(1)}$ in $\mathcal{D}_1$, and $y^{(2)}$ in $\mathcal{D}_2$ can be obtained in the similar way as follows:

$$y^{(1)} = p^{(1)} + q^{(1)}, \quad y^{(2)} = p^{(2)} + q^{(2)}. \tag{7}$$

For simplicity, we redefine all wights and biases in the network as $\boldsymbol{\Phi}$ and the input data in all domains as $\boldsymbol{X}$:

$$\begin{aligned}\boldsymbol{W} &= \{\boldsymbol{W}^{(i,j)}\}_{i=1,\cdots,6, j=1,\cdots,k(i)} \cup \{\boldsymbol{W}^{p^{(z)}}, \boldsymbol{W}^{q^{(z)}}\}_{z=1,2},\\ \boldsymbol{b} &= \{\boldsymbol{b}^{(i,j)}\}_{i=1,\cdots,6, j=1,\cdots,k(i)} \cup \{\boldsymbol{b}^{p^{(z)}}, \boldsymbol{b}^{q^{(z)}}\}_{z=1,2},\\ \boldsymbol{\Phi} &= \boldsymbol{W} \cup \boldsymbol{b}, \quad \boldsymbol{X} = \{\boldsymbol{X}^{(i)}\}_{i=1,\cdots,6}.\end{aligned} \tag{8}$$

The objective is to make the predicted rating value $y^{(1)}/y^{(2)}$ close to the real rating value $r^{(1)}/r^{(2)}$, so we define the likelihood function as follows:

$$p(\boldsymbol{y}^{(1)}, \boldsymbol{y}^{(2)}|\boldsymbol{X}, \boldsymbol{\Phi}, \delta_0, \gamma_0) = \prod_{t=1}^{T^{(1)}} \mathcal{N}(r_t^{(1)}|y_t^{(1)}, \delta_0^2) \prod_{t=1}^{T^{(2)}} \mathcal{N}(r_t^{(2)}|y_t^{(2)}, \delta_0^2), \tag{9}$$

Finally, we can get the posterior distribution of $\boldsymbol{\Phi}$ as follows:

$$p(\boldsymbol{\Phi}|\boldsymbol{X}, \boldsymbol{y}^{(1)}, \boldsymbol{y}^{(2)}, \delta_0, \gamma_0) = \frac{p(\boldsymbol{y}^{(1)}, \boldsymbol{y}^{(2)}|\boldsymbol{X}, \boldsymbol{\Phi}, \delta_0, \gamma_0) p(\boldsymbol{\Phi}|\gamma_0)}{p(\boldsymbol{y}^{(1)}, \boldsymbol{y}^{(2)}|\boldsymbol{X})} \tag{10}$$

where $p(\boldsymbol{y}^{(1)}, \boldsymbol{y}^{(2)}|\boldsymbol{X})$ is a normalization constant.

However, the function is too complex to lend itself to exact Bayesian Inference. So, it is difficult to calculate the posterior distribution directly. To address this problem, we introduce BBB to find the approximate posterior distribution. BBB adopts variational inference as optimization which finds the parameters of a approximate posterior distribution on the weights by minimizing the Kullback-Leibler (KL) divergence with the true Bayesian posterior. $q(\Phi|\theta)$ denotes the approximate posterior distribution of Φ where θ is the parameter of the approximate posterior distribution. We suppose the approximate posterior distribution is still a Gaussian distribution, in this case, $\theta = (\mu, \sigma)$ denotes the mean and the standard deviation of the Gaussian distribution. Therefore, the optimal θ^* can be obtained by optimizing the following objective:

$$\begin{aligned}\theta^* &= \arg\min_{\theta} \quad \mathrm{KL}[q(\boldsymbol{\Phi}|\theta)||p(\boldsymbol{\Phi}|\boldsymbol{X}, \boldsymbol{y}^{(1)}, \boldsymbol{y}^{(2)}, \delta_0, \gamma_0)] \\ &= \arg\min_{\theta} \int q(\boldsymbol{\Phi}|\theta) \log \frac{q(\boldsymbol{\Phi}|\theta)}{p(\boldsymbol{\Phi}|\gamma_0) p(\boldsymbol{y}^{(1)}, \boldsymbol{y}^{(2)}|\boldsymbol{\Phi}, \boldsymbol{X}, \delta_0)} \mathrm{d}\boldsymbol{\Phi} \\ &= \arg\min_{\theta} \quad \mathbb{E}_{q(\boldsymbol{\Phi}|\theta)}[\log q(\boldsymbol{\Phi}|\theta) - \log p(\boldsymbol{\Phi}|\gamma_0)] \\ &\qquad - \mathbb{E}_{q(\boldsymbol{\Phi}|\theta)}[\log p(\boldsymbol{y}^{(1)}, \boldsymbol{y}^{(2)}|\boldsymbol{\Phi}, \boldsymbol{X}, \delta_0)].\end{aligned} \tag{11}$$

However, we can't calculate the derivative of the optimization function (11) because the derivative of an expectation can't be calculated directly. According to [26], we parameterize the σ as $\sigma = \log(1 + \exp(\rho))$ and then σ will always be non-negative. The parameters of the approximate posterior distribution can be denoted as $\theta = (\mu, \rho)$. The objective function (11) can be re-written as the following formulation:

$$\begin{aligned}H(\boldsymbol{\Phi}|\theta) &= \log q(\boldsymbol{\Phi}|\theta) - \log p(\boldsymbol{\Phi}|\gamma_0) - \log p(\boldsymbol{y}^{(1)}, \boldsymbol{y}^{(2)}|\boldsymbol{\Phi}, \boldsymbol{X}, \delta_0), \\ \theta^* &= \arg\min_{\theta} \quad \mathbb{E}_{q(\boldsymbol{\Phi}|\theta)}[H(\boldsymbol{\Phi}|\theta)].\end{aligned} \tag{12}$$

Then we can get the objective function above is approximately equal to the following function:

$$\begin{aligned}&\frac{\partial \mathbb{E}_{H(\boldsymbol{\Phi}|\theta)}[H(\boldsymbol{\Phi}, \theta)]}{\partial \theta} \\ &\approx \frac{1}{S} \frac{\partial H(c(\theta, \epsilon^{(s)}), \theta)}{\partial c(\theta, \epsilon^{(s)})} \frac{\partial c(\theta, \epsilon^{(s)})}{\partial \theta} + \frac{\partial H(c(\theta, \epsilon^{(s)}), \theta)}{\partial \theta},\end{aligned} \tag{13}$$

where $\epsilon^{(s)}$ denotes the Monte Carlo sample drawn from the distribution $q(\epsilon^{(s)})$.

IV. Experiments

In this section, we evaluate our proposed model GCBAN on real-world datasets and compare it with several state-of-the-art cross-domain algorithms and single domain methods.

A. Data Description

There are three series of datasets in our experiments: Movie100k, Movie1m and Douban, and the detailed information of these three datasets is summarized in Table II.

Movie100k and Movie1m are two open datasets from GroupLens which include user features, user id, item features, item id and ratings. The rating ranges from 1 to 5. The user features contains user's age, gender, profession, etc. And item feature includes item's genres. Id information is encoded into a vector by one-hot encoding. We further split Movie100k and Movie1m into two domains according to the type of movie. Movies with high genre similarity will be divided into a same domain. Movie100k-D1 (Movie100k-D2) represents the domain 1 (domain 2) from Movie100k, and Movie1m-D1 and Movie1m-D2 are named similarly.

Douban is a well known Chinese social network platform. The Douban dataset includes three recommendation tasks like movie (Douban-Movie), music (Douban-Music) and book (Douban-Book) recommendations. The user's features includes gender, age, place of residence, tag, etc. The features of movies consist of movie's name, director, issuer, genre, language, etc. Musics' features consist of music's name, singer's name, genre, publisher, etc. The features of books include publisher country, category, etc.

Finally, we construct eight cross-domain recommendation tasks: Movie100k-D1 $\rightleftharpoons$ Movie100k-D2; Movie1m-D1 $\rightleftharpoons$ Movie1m-D2; Douban-Movie $\rightleftharpoons$ Douban-Music; Douban-Music $\rightleftharpoons$ Douban-Book. We train two domains simultaneously and evaluate the test data from these two domains. Douban-Music has two results because it exists in two tasks, and we only report the best results for all algorithms.

TABLE II: Statistics of datasets.

Dataset	Movie100k-D1	Movie100k-D2	Movie1m-D1	Movie1m-D2	Douban-Movie	Douban-Music	Douban-Book
Users	925	925	5,959	5,959	13,909	13,909	13,909
Items	729	950	1,659	2,044	57,199	49,659	28,163
Rating	52,545	45,369	535,896	448,093	1,220,836	191,283	106,189
Density	0.0779	0.0516	0.0542	0.0368	0.0015	0.0003	0.0003

B. Competitors and Implementation Details

For evaluating, we use Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE), which are the most commonly used evaluation metrics in recommendation systems. And we compare our model GCBAN with four cross-domain recommendation competitors and two single-domain recommendation baselines.

- MVDNN [23]: it is a deep learning based cross-domain recommendation system to map users' and items' auxiliary information to a latent space where the similarity between users and their preferred items is maximized.
- CCCFNet [24]: it is a cross-domain recommendation system which combines collaborative filtering and content-based filtering in a unified framework. But it only introduce the auxiliary information from items without users.
- NeuCDCF[1] : it is a neural cross-domain collaborative filtering without introducing auxiliary information.
- CLFM [27]: it is a cross-domain recommendation algorithm via cluster-level latent factor model.
- DeepMF: it is a novel matrix factorization model with neural network architecture [5].
- PMF: it is a probabilistic matrix factorization model [4].

GCBAN is implemented based on Tensorflow. We use standard gaussian distribution to initialize parameters μ and ρ and use the optimization method Adam with learning rate 0.0005. Hyper-parameters γ_0 and δ_0 are both sampled from

[1]https://github.com/mvijaikumar/NeuCDCF.

the integer set {1e-5, 1e-4, 1e-4, 1e-3, 1e-2, 1e-1, 1e-0}, the dimension of latent representations d^a for user and item attributes and the dimension of latent representations d^o for user and item one-hot codings are both sampled from the integer set {5, 10, 15, 20, 25}. Since the dimension of items' attributes is almost the same size among different domains, we set the dimension of items' attributes from all domains the same. And the Monte Carlo sample times S is set to 3. For NeuCDCF, we set the parameters and the structure of the network as the default setting in the source code. For MVDNN and CCCFNet, we adopt two-layer networks as the same with our model GCBAN. For DeepMF, we adopt two-layer networks as suggested in its original paper. The dimension of the latent factor is chosen from the integer set {8, 16, 32, 64} for each dataset according to its original paper. For CLFM, the number of user and item clusters K and L in both domains are set as K = 30 and L = 50 and the dimension of shared common space is set to 40 which are suggested according to its original paper. Finally, the dimension of the latent factor is chosen from the integer set {8, 16, 32, 64} for PMF. Because CLFM needs to learn the shared cluster-level user-item rating patterns firstly which demands a lot of time due to the high computation complexity of cluster algorithms. So when deal with the large-scale data which have many users and items like Douban-Music and Douban-Book, CLFM returns no result within 24 hours. Thus there is no result for CLFM in Douban.

C. Normal Experiments

In the normal experiments, we randomly divide the target domain data into two parts where one for training and the rest for testing. Specifically, we respectively sample 80%, 60%, 40% and 20% for training and the rest for test. The average performance on five individual trials is recorded, and all the results on three datasets are shown in Table III and Figure 3. From these results, we have the following insightful observations:

- GCBAN outperforms PMF and DeepMF, which indicates that cross-domain method can take advantage of information from related source domains.
- Neural network based cross-domain recommendation systems NeuCDCF, CCCFNet and GCBAN perform better than the traditional cross-domain recommendation system CLFM. The reason may be that neural network can learn powerful representations. CCCFNet and GCBAN are better than cross-domain recommendations like CLFM and NeuCDCF on most datasets. The reason may be that they introduce auxiliary information to enhance the recommendation performance. Also, we observe that MVDNN can not obtain satisfying performance on most datasets. This might be the reason that although MVDNN only learns the auxiliary information by neural networks and does not take the collaborative filtering into account to model the relevance between users and items.
- GCBAN achieves the best performance on these three datasets. On the one hand, GCBAN performs better than CLFM and NeuCDCF where we attribute to that GCBAN learn the model from all the domains simultaneously and these domains can complement each other. On the other hand, GCBAN outperforms better than NeuCDCF, MVDNN and CCCDNet. This may be the reasons, 1) GCBAN takes the advantages of the neural network and Bayesian framework which introduces the uncertainty to models; 2) GCBAN infers an approximate posterior under the Bayesian framework instead of a point estimate, and makes more robust predictions with Bayesian model averaging over the posterior.
- The sampled proportions for training are {0.8, 0.6, 0.4, 0.2}. Suppose the density of the original rating matrix is α, it means that we do experiments with different sparsity {1-0.8α, 1-0.6α, 1-0.4α, 1-0.2α}. Overall, GCBAN can achieve the best results under different training sparsity, which again validates the effectiveness of GCBAN.

TABLE III: The Detailed Results on Movie100k and Movie1m

Dataset	Training	Metrics	PMF	DeepMF	CLFM	NeuCDCF	MVDNN	CCCFNet	GCBAN
Movie 100k-D1	80%	RMSE	0.9718	0.9278	1.0821	0.9209	1.1135	0.9643	**0.9042**
		MAE	0.7523	0.7311	0.8904	0.7183	0.9257	0.7735	**0.7137**
	60%	RMSE	1.0294	0.9379	1.0966	0.9329	1.1178	0.9838	**0.9256**
		MAE	0.7990	0.7381	0.9132	0.7268	0.9340	0.7928	**0.7312**
	40%	RMSE	1.1060	0.9500	1.1110	0.9554	1.1200	1.0119	**0.9391**
		MAE	0.8600	0.7469	0.9288	0.7452	0.9432	0.8232	**0.7433**
	20%	RMSE	1.1923	0.9913	1.1195	1.0166	1.1275	1.0497	**0.9610**
		MAE	0.9301	0.7784	0.9367	0.7938	0.9471	0.8667	**0.7674**
Movie 100k-D2	80%	RMSE	1.0425	0.9580	1.0981	0.9616	1.1076	0.9782	**0.9291**
		MAE	0.8133	0.7544	0.9033	0.7572	0.9162	0.7843	**0.7333**
	60%	RMSE	1.0975	0.9633	1.1003	0.9726	1.1094	0.9924	**0.9486**
		MAE	0.8557	0.7591	0.9058	0.7653	0.9239	0.7996	**0.7493**
	40%	RMSE	1.1736	0.9812	1.1095	0.9975	1.1130	1.0191	**0.9613**
		MAE	0.9152	0.7728	0.9200	0.7857	0.9273	0.8292	**0.7611**
	20%	RMSE	1.2171	1.0339	1.1103	1.0569	1.1209	1.0543	**0.9830**
		MAE	0.9544	0.8160	0.9216	0.8316	0.9379	0.8696	**0.7828**
Movie 1m-D1	80%	RMSE	0.8719	0.8860	1.094	0.8825	1.0783	0.8676	**0.8502**
		MAE	0.6781	0.6953	0.6836	0.9115	0.8759	0.6785	**0.6668**
	60%	RMSE	0.8924	0.9221	1.1092	0.8873	1.0852	0.8754	**0.8631**
		MAE	0.6948	0.7207	0.9278	0.6875	0.8826	0.6881	**0.6776**
	40%	RMSE	0.9172	0.9379	1.1211	0.8985	1.0856	0.8938	**0.8796**
		MAE	0.7156	0.7381	0.9394	0.6976	0.8806	0.7088	**0.6924**
	20%	RMSE	0.9588	0.9500	1.1306	0.9273	1.0981	0.9272	**0.9051**
		MAE	0.7524	0.7469	0.9487	0.7229	0.9047	0.7376	**0.7131**
Movie 1m-D2	80%	RMSE	0.9024	0.9098	1.0846	0.9193	1.0740	0.8815	**0.8675**
		MAE	0.7048	0.7151	0.8981	0.7171	0.8747	0.6949	**0.6815**
	60%	RMSE	0.9159	0.9219	1.0856	0.929	1.0743	0.8970	**0.8833**
		MAE	0.7171	0.7265	0.8987	0.7231	0.8791	0.7065	**0.6948**
	40%	RMSE	0.9390	0.9285	1.0868	0.9360	1.0754	0.9113	**0.9018**
		MAE	0.7381	0.7294	0.8997	0.7317	0.8809	0.7192	**0.7104**
	20%	RMSE	1.0112	0.9553	1.0889	0.9652	1.0751	0.9411	**0.9289**
		MAE	0.7944	0.7490	0.9012	0.7581	0.8887	0.7463	**0.7333**

D. Cold-start Experiments

To evaluate the prediction performance of recommendation methods when they address the cold-start problem, we conduct cold-start experiments. We respectively extract users and items with corresponding feature information and rating records from dataset and regard them as test datasets. The split proportion of users and items is sampled from {0.2, 0.4, 0.6, 0.8}, and the results of RMSE are shown in Figure 4.

As we can see from Figure 4, GCBAN performs better than MVDNN and CCCFNet. This is because our model incorporates the auxiliary information from both users and items and captures the uncertainty of the network. Especially, we can find that GCBAN performs better than CCCFNet in the cold-start problem of users. The reason may be that

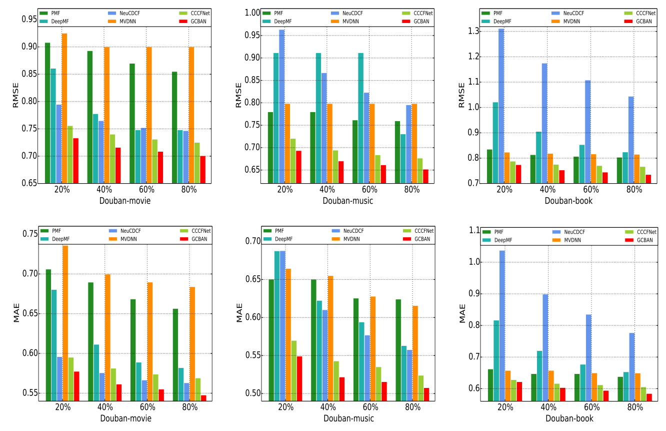


Fig. 3: The Comparison Results on Douban Dataset

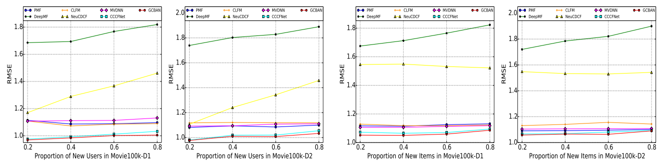


Fig. 4: New User and Item Problems

CCCFNet only introduces the auxiliary information of items, thus it can not deal with cold-start problem of users. Also, we can find GCBAN performs better than the single-domain recommendation ones. We attribute it to that cross-domain method can help transfer the knowledge from related source domain and enhance the recommendation quality.

V. Conclusion

To flexibly integrate as much information as possible from multiple sources for recommendation, we proposed a general cross-domain framework via Bayesian neural network. In other word, our model not only incorporates the auxiliary information from both users and items but also transfers knowledge from related source domains, which can take advantage of both hybrid recommendation approach and cross-domain method. Note that, the latent representations of users learnt from ratings and auxiliary information are shared across different domains for knowledge transfer. Moreover, we introduced uncertainty to all weights which are represented by probability distributions in our neural networks to make calibrated probabilistic predictions and avoid over-fitting. Finally, we conducted extensive experiments on real-world data sets to demonstrate the effectiveness of our model.

Acknowledgment

The research work is supported by the National Key Research and Development Program of China under Grant No. 2018YFB1004300, the National Natural Science Foundation of China under Grant No. 61773361, 61473273, 91546122, Guangdong provincial science and technology plan projects under Grant No. 2015 B010109005, the Project of Youth Innovation Promotion Association CAS under Grant No. 2017146. This work is also partly supported by the funding of WeChat cooperation project. We thank Bo Chen, Leyu Lin, Xiaohu Cheng for their constructive advices.

References

[1] G. Adomavicius and A. Tuzhilin, *Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions*. Springer International Publishing, 2013.

[2] T. Man, H. Shen, X. Jin, and X. Cheng, "Cross-domain recommendation: An embedding and mapping approach," in *IJCAI*, 2017, pp. 2464–2470.

[3] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "Bpr: Bayesian personalized ranking from implicit feedback," in *UAI*. AUAI Press, 2009, pp. 452–461.

[4] A. Mnih and R. R. Salakhutdinov, "Probabilistic matrix factorization," in *NIPS*, 2008, pp. 1257–1264.

[5] H. J. Xue, X. Dai, J. Zhang, S. Huang, and J. Chen, "Deep matrix factorization models for recommender systems," in *IJCAI*, 2017, pp. 3203–3209.

[6] Y. Zhou, D. Wilkinson, R. Schreiber, and R. Pan, "Large-scale parallel collaborative filtering for the netflix prize," in *Proc. Int'l Conf. Algorithmic Aspects in Information and Management, Lncs*, 2008, pp. 337–348.

[7] A. P. Singh and G. J. Gordon, "Relational learning via collective matrix factorization," in *SIGKDD*. ACM, 2008, pp. 650–658.

[8] H. Ma, D. Zhou, C. Liu, M. R. Lyu, and I. King, "Recommender systems with social regularization," in *WSDM*. ACM, 2011, pp. 287–296.

[9] A. P. Singh and G. Gordon, "A bayesian matrix factorization model for relational data," *Computer Science*, pp. 556–563, 2010.

[10] Y. Gao, X. Wang, G. Lei, and Z. Chen, "Learning to recommend with collaborative matrix factorization for new users," *Journal of Computer Research and Development*, 2017.

[11] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[12] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *science*, vol. 313, no. 5786, pp. 504–507, 2006.

[13] S. Li, J. Kawale, and Y. Fu, "Deep collaborative filtering via marginalized denoising auto-encoder," in *CIKM*. ACM, 2015, pp. 811–820.

[14] H. Wang, N. Wang, and D.-Y. Yeung, "Collaborative deep learning for recommender systems," in *SIGKDD*. ACM, 2015, pp. 1235–1244.

[15] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *WWW*. International World Wide Web Conferences Steering Committee, 2017, pp. 173–182.

[16] X. Wang and Y. Wang, "Improving content-based and hybrid music recommendation using deep learning," in *MM*. ACM, 2014, pp. 627–636.

[17] Y. Zhang, B. Cao, and D. Y. Yeung, "Multi-domain collaborative filtering," in *UAI*, 2010, pp. 725–732.

[18] B. Li, Q. Yang, and X. Xue, "Can movies and books collaborate? cross-domain collaborative filtering for sparsity reduction." in *IJCAI*, 2009, pp. 2052–2057.

[19] B. Li, Q. Yang, and X. Y. Xue, "Transfer learning for collaborative filtering via a rating-matrix generative model," in *ICML 2009*, 2009, pp. 617–624.

[20] W. Pan, E. W. Xiang, N. N. Liu, and Q. Yang, "Transfer learning in collaborative filtering for sparsity reduction." in *AAAI*, 2012, pp. 662–668.

[21] Y. F. Liu, C. Y. Hsu, and S. H. Wu, "Non-linear cross-domain collaborative filtering via hyper-structure transfer," in *ICML*, 2015, pp. 1190–1198.

[22] D. Agarwal, B. C. Chen, and B. Long, "Localized factor models for multi-context recommendation," in *SIGKDD*, 2011, pp. 609–617.

[23] A. M. Elkahky, Y. Song, and X. He, "A multi-view deep learning approach for cross domain user modeling in recommendation systems," *WWW*, pp. 278–288, 2015.

[24] J. Lian, F. Zhang, X. Xie, and G. Sun, "Cccfnet: A content-boosted collaborative filtering neural network for cross domain recommender systems," in *WWW*, 2017, pp. 817–818.

[25] J. M. Hernández-Lobato and R. Adams, "Probabilistic backpropagation for scalable learning of bayesian neural networks," in *ICML*, 2015, pp. 1861–1869.

[26] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight uncertainty in neural networks," in *ICML*, 2015, pp. 1613–1622.

[27] S. Gao, H. Luo, D. Chen, S. Li, P. Gallinari, and J. Guo, "Cross-domain recommendation via cluster-level latent factor model," in *ECML-PKDD*, 2013, pp. 161–176.