Collaborative Deep Learning for Recommender Systems

Hao Wang
Hong Kong University of Science and Technology
hwangaz@cse.ust.hk

Naiyan Wang
Hong Kong University of Science and Technology
winsty@gmail.com

Dit-Yan Yeung
Hong Kong University of Science and Technology
dyyeung@cse.ust.hk

ABSTRACT

Collaborative filtering (CF) is a successful approach commonly used by many recommender systems. Conventional CF-based methods use the ratings given to items by users as the sole source of information for learning to make recommendation. However, the ratings are often very sparse in many applications, causing CF-based methods to degrade significantly in their recommendation performance. To address this sparsity problem, auxiliary information such as item content information may be utilized. Collaborative topic regression (CTR) is an appealing recent method taking this approach which tightly couples the two components that learn from two different sources of information. Nevertheless, the latent representation learned by CTR may not be very effective when the auxiliary information is very sparse. To address this problem, we generalize recent advances in deep learning from i.i.d. input to non-i.i.d. (CF-based) input and propose in this paper a hierarchical Bayesian model called collaborative deep learning (CDL), which jointly performs deep representation learning for the content information and collaborative filtering for the ratings (feedback) matrix. Extensive experiments on three real-world datasets from different domains show that CDL can significantly advance the state of the art.

Categories and Subject Descriptors

H.1.0 [**Information Systems**]: Models and Principles—*General*; J.4 [**Computer Applications**]: Social and Behavioral Sciences

Keywords

Recommender systems; Deep learning; Topic model; Text mining

1. INTRODUCTION

Due to the abundance of choice in many online services, recommender systems (RS) now play an increasingly significant role [40]. For individuals, using RS allows us to make more effective use of information. Besides, many companies (e.g., Amazon and Netflix) have been using RS extensively to target their customers by recommending products or services. Existing methods for RS can roughly be categorized into three classes [6]: content-based methods, collaborative filtering (CF) based methods, and hybrid methods. Content-based methods [17] make use of user profiles or product descriptions for recommendation. CF-based methods [23, 27] use the past activities or preferences, such as user ratings on items, without using user or product content information. Hybrid methods [1, 18, 12] seek to get the best of both worlds by combining content-based and CF-based methods.

Because of privacy concerns, it is generally more difficult to collect user profiles than past activities. Nevertheless, CF-based methods do have their limitations. The prediction accuracy often drops significantly when the ratings are very sparse. Moreover, they cannot be used for recommending new products which have yet to receive rating information from users. Consequently, it is inevitable for CF-based methods to exploit auxiliary information and hence hybrid methods have gained popularity in recent years.

According to whether two-way interaction exists between the rating information and auxiliary information, we may further divide hybrid methods into two sub-categories: loosely coupled and tightly coupled methods. Loosely coupled methods like [29] process the auxiliary information once and then use it to provide features for the CF models. Since information flow is one-way, the rating information cannot provide feedback to guide the extraction of useful features. For this sub-category, improvement often has to rely on a manual and tedious feature engineering process. On the contrary, tightly coupled methods like [34] allow two-way interaction. On one hand, the rating information can guide the learning of features. On the other hand, the extracted features can further improve the predictive power of the CF models (e.g., based on matrix factorization of the sparse rating matrix). With two-way interaction, tightly coupled methods can automatically learn features from the auxiliary information and naturally balance the influence of the rating and auxiliary information. This is why tightly coupled methods often outperform loosely coupled ones [35].

Collaborative topic regression (CTR) [34] is a recently proposed tightly coupled method. It is a probabilistic graphical model that seamlessly integrates a topic model, latent Dirichlet allocation (LDA) [5], and a model-based CF method, probabilistic matrix factorization (PMF) [27]. CTR is an

KDD'15, August 10-13, 2015, Sydney, NSW, Australia.

DOI: http://dx.doi.org/10.1145/2783258.2783273.

appealing method in that it produces promising and interpretable results. Nevertheless, the latent representation learned is often not effective enough especially when the auxiliary information is very sparse. It is this representation learning problem that we will focus on in this paper.

On the other hand, deep learning models recently show great potential for learning effective representations and deliver state-of-the-art performance in computer vision [38] and natural language processing [15, 26] applications. In deep learning models, features are learned in a supervised or unsupervised manner. Although they are more appealing than shallow models in that the features can be learned automatically (e.g., effective feature representation is learned from text content), they are inferior to shallow models such as CF in capturing and learning the similarity and implicit relationship between items. This calls for integrating deep learning with CF by performing deep learning collaboratively.

Unfortunately, very few attempts have been made to develop deep learning models for CF. [28] uses restricted Boltzmann machines instead of the conventional matrix factorization formulation to perform CF and [9] extends this work by incorporating user-user and item-item correlations. Although these methods involve both deep learning and CF, they actually belong to CF-based methods because they do not incorporate content information like CTR, which is crucial for accurate recommendation. [24] uses low-rank matrix factorization in the last weight layer of a deep network to significantly reduce the number of model parameters and speed up training, but it is for classification instead of recommendation tasks. On music recommendation, [21, 39] directly use conventional CNN or deep belief networks (DBN) to assist representation learning for content information, but the deep learning components of their models are deterministic without modeling the noise and hence they are less robust. The models achieve performance boost mainly by loosely coupled methods without exploiting the interaction between content information and ratings. Besides, the CNN is linked directly to the rating matrix, which means the models will perform poorly when the ratings are sparse, as shown in the following experiments.

To address the challenges above, we develop a hierarchical Bayesian model called collaborative deep learning (CDL) as a novel tightly coupled method for RS. We first present a Bayesian formulation of a deep learning model called stacked denoising autoencoder (SDAE) [32]. With this, we then present our CDL model which tightly couples deep representation learning for the content information and collaborative filtering for the ratings (feedback) matrix, allowing two-way interaction between the two. Experiments show that CDL significantly outperforms the state of the art. Note that although we present CDL as using SDAE for its feature learning component, CDL is actually a more general framework which can also admit other deep learning models such as deep Boltzmann machines [25], recurrent neural networks [10], and convolutional neural networks [16].

The main contribution of this paper is summarized below:

- By performing deep learning collaboratively, CDL can simultaneously extract an effective deep feature representation from content and capture the similarity and implicit relationship between items (and users). The learned representation may also be used for tasks other than recommendation.
- Unlike previous deep learning models which use simple target like classification [15] and reconstruction [32], we propose to use CF as a more complex target in a probabilistic framework.
- Besides the algorithm for attaining maximum a posteriori (MAP) estimates, we also derive a sampling-based algorithm for the Bayesian treatment of CDL, which, interestingly, turns out to be a Bayesian generalized version of back-propagation.
- To the best of our knowledge, CDL is the first hierarchical Bayesian model to bridge the gap between state-of-the-art deep learning models and RS. Besides, due to its Bayesian nature, CDL can be easily extended to incorporate other auxiliary information to further boost the performance.
- Extensive experiments on three real-world datasets from different domains show that CDL can significantly advance the state of the art.

2. NOTATION AND PROBLEM FORMULATION

Similar to the work in [34], the recommendation task considered in this paper takes implicit feedback [13] as the training and test data. The entire collection of J items (articles or movies) is represented by a J-by-S matrix $\mathbf{X}_c$, where row j is the bag-of-words vector $\mathbf{X}_{c,j*}$ for item j based on a vocabulary of size S. With I users, we define an I-by-J binary rating matrix $\mathbf{R} = [\mathbf{R}_{ij}]_{I \times J}$. For example, in the dataset *citeulike-a* $\mathbf{R}_{ij} = 1$ if user i has article j in his or her personal library and $\mathbf{R}_{ij} = 0$ otherwise. Given part of the ratings in $\mathbf{R}$ and the content information $\mathbf{X}_c$, the problem is to predict the other ratings in $\mathbf{R}$. Note that although we focus on movie recommendation (where plots of movies are considered as content information) and article recommendation like [34] in this paper, our model is general enough to handle other recommendation tasks (e.g., tag recommendation).

The matrix $\mathbf{X}_c$ plays the role of clean input to the SDAE while the noise-corrupted matrix, also a J-by-S matrix, is denoted by $\mathbf{X}_0$. The output of layer l of the SDAE is denoted by $\mathbf{X}_l$ which is a J-by-K_l matrix. Similar to $\mathbf{X}_c$, row j of $\mathbf{X}_l$ is denoted by $\mathbf{X}_{l,j*}$. $\mathbf{W}_l$ and $\mathbf{b}_l$ are the weight matrix and bias vector, respectively, of layer l, $\mathbf{W}_{l,*n}$ denotes column n of $\mathbf{W}_l$, and L is the number of layers. For convenience, we use $\mathbf{W}^+$ to denote the collection of all layers of weight matrices and biases. Note that an $L/2$-layer SDAE corresponds to an L-layer network.

3. COLLABORATIVE DEEP LEARNING

We are now ready to present details of our CDL model. We first briefly review SDAE and give a Bayesian formulation of SDAE. This is then followed by the presentation of CDL as a hierarchical Bayesian model which tightly integrates the ratings and content information.

3.1 Stacked Denoising Autoencoders

SDAE [32] is a feedforward neural network for learning representations (encoding) of the input data by learning to predict the clean input itself in the output, as shown in Figure 2. Usually the hidden layer in the middle, i.e., $\mathbf{X}_2$ in the figure, is constrained to be a bottleneck and the input layer $\mathbf{X}_0$ is a corrupted version of the clean input data. An

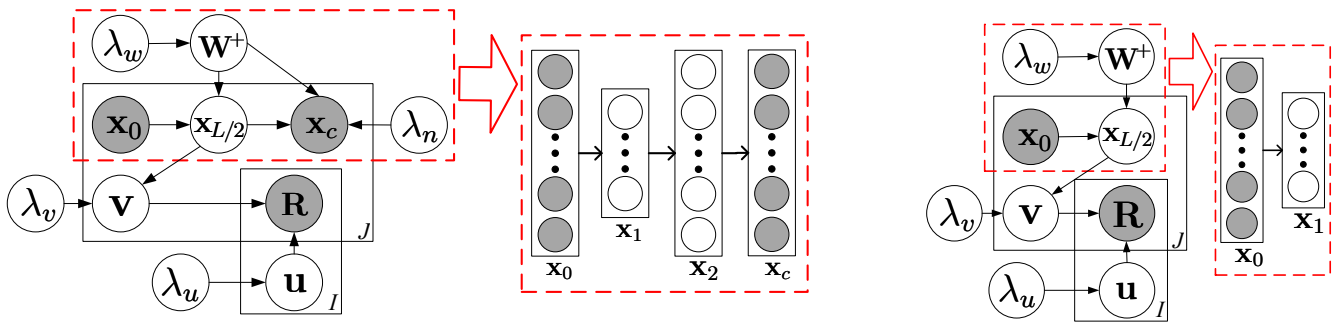

Figure 1: On the left is the graphical model of CDL. The part inside the dashed rectangle represents an SDAE. An example SDAE with $L = 2$ is shown. On the right is the graphical model of the degenerated CDL. The part inside the dashed rectangle represents the encoder of an SDAE. An example SDAE with $L = 2$ is shown on the right of it. Note that although L is still 2, the decoder of the SDAE vanishes. To prevent clutter, we omit all variables $\mathbf{x}_l$ except $\mathbf{x}_0$ and $\mathbf{x}_{L/2}$ in the graphical models.

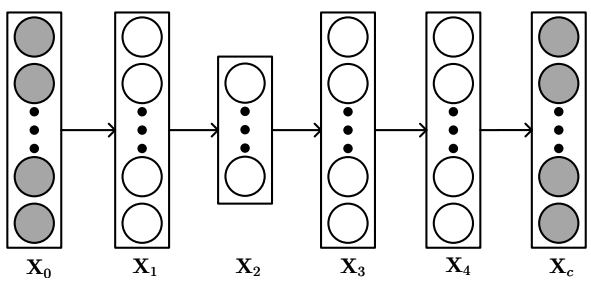

Figure 2: A 2-layer SDAE with $L = 4$.

SDAE solves the following optimization problem:

$$\min_{\{\mathbf{W}_l\},\{\mathbf{b}_l\}} \|\mathbf{X}_c - \mathbf{X}_L\|_F^2 + \lambda \sum_l \|\mathbf{W}_l\|_F^2,$$

where λ is a regularization parameter and $\|\cdot\|_F$ denotes the Frobenius norm.

3.2 Generalized Bayesian SDAE

If we assume that both the clean input $\mathbf{X}_c$ and the corrupted input $\mathbf{X}_0$ are observed, similar to [4, 19, 3, 7], we can define the following generative process:

1. For each layer l of the SDAE network,
   (a) For each column n of the weight matrix $\mathbf{W}_l$, draw
   $$\mathbf{W}_{l,*n} \sim \mathcal{N}(\mathbf{0}, \lambda_w^{-1}\mathbf{I}_{K_l}).$$
   (b) Draw the bias vector $\mathbf{b}_l \sim \mathcal{N}(\mathbf{0}, \lambda_w^{-1}\mathbf{I}_{K_l})$.
   (c) For each row j of $\mathbf{X}_l$, draw
   $$\mathbf{X}_{l,j*} \sim \mathcal{N}(\sigma(\mathbf{X}_{l-1,j*}\mathbf{W}_l + \mathbf{b}_l), \lambda_s^{-1}\mathbf{I}_{K_l}). \quad (1)$$
2. For each item j, draw a clean input [1]
   $$\mathbf{X}_{c,j*} \sim \mathcal{N}(\mathbf{X}_{L,j*}, \lambda_n^{-1}\mathbf{I}_J).$$

Note that if λ_s goes to infinity, the Gaussian distribution in Equation (1) will become a Dirac delta distribution [31] centered at $\sigma(\mathbf{X}_{l-1,j*}\mathbf{W}_l + \mathbf{b}_l)$, where $\sigma(\cdot)$ is the sigmoid function. The model will degenerate to be a Bayesian formulation of SDAE. That is why we call it generalized SDAE.

Note that the first $L/2$ layers of the network act as an encoder and the last $L/2$ layers act as a decoder. Maximization of the posterior probability is equivalent to minimization of the reconstruction error with weight decay taken into consideration.

[1]Note that while generation of the *clean* input $\mathbf{X}_c$ from $\mathbf{X}_L$ is part of the generative process of the Bayesian SDAE, generation of the *noise-corrupted* input $\mathbf{X}_0$ from $\mathbf{X}_c$ is an artificial noise injection process to help the SDAE learn a more robust feature representation.

3.3 Collaborative Deep Learning

Using the Bayesian SDAE as a component, the generative process of CDL is defined as follows:

1. For each layer l of the SDAE network,
   (a) For each column n of the weight matrix $\mathbf{W}_l$, draw
   $$\mathbf{W}_{l,*n} \sim \mathcal{N}(\mathbf{0}, \lambda_w^{-1}\mathbf{I}_{K_l}).$$
   (b) Draw the bias vector $\mathbf{b}_l \sim \mathcal{N}(\mathbf{0}, \lambda_w^{-1}\mathbf{I}_{K_l})$.
   (c) For each row j of $\mathbf{X}_l$, draw
   $$\mathbf{X}_{l,j*} \sim \mathcal{N}(\sigma(\mathbf{X}_{l-1,j*}\mathbf{W}_l + \mathbf{b}_l), \lambda_s^{-1}\mathbf{I}_{K_l}).$$
2. For each item j,
   (a) Draw a clean input $\mathbf{X}_{c,j*} \sim \mathcal{N}(\mathbf{X}_{L,j*}, \lambda_n^{-1}\mathbf{I}_J)$.
   (b) Draw a latent item offset vector $\boldsymbol{\epsilon}_j \sim \mathcal{N}(\mathbf{0}, \lambda_v^{-1}\mathbf{I}_K)$ and then set the latent item vector to be:
   $$\mathbf{v}_j = \boldsymbol{\epsilon}_j + \mathbf{X}_{\frac{L}{2},j*}^T.$$
3. Draw a latent user vector for each user i:
   $$\mathbf{u}_i \sim \mathcal{N}(\mathbf{0}, \lambda_u^{-1}\mathbf{I}_K).$$
4. Draw a rating $\mathbf{R}_{ij}$ for each user-item pair (i, j):
   $$\mathbf{R}_{ij} \sim \mathcal{N}(\mathbf{u}_i^T\mathbf{v}_j, \mathbf{C}_{ij}^{-1}).$$

Here λ_w, λ_n, λ_u, λ_s, and λ_v are hyperparameters and $\mathbf{C}_{ij}$ is a confidence parameter similar to that for CTR ($\mathbf{C}_{ij} = a$ if $\mathbf{R}_{ij} = 1$ and $\mathbf{C}_{ij} = b$ otherwise). Note that the middle layer $\mathbf{X}_{L/2}$ serves as a bridge between the ratings and content information. This middle layer, along with the latent offset $\boldsymbol{\epsilon}_j$, is the key that enables CDL to simultaneously learn an effective feature representation and capture the similarity and (implicit) relationship between items (and users). Similar to the generalized SDAE, for computational efficiency, we can also take λ_s to infinity.

The graphical model of CDL when λ_s approaches positive infinity is shown in Figure 1, where, for notational simplicity, we use $\mathbf{x}_0$, $\mathbf{x}_{L/2}$, and $\mathbf{x}_L$ in place of $\mathbf{X}_{0,j*}^T$, $\mathbf{X}_{\frac{L}{2},j*}^T$, and $\mathbf{X}_{L,j*}^T$, respectively.

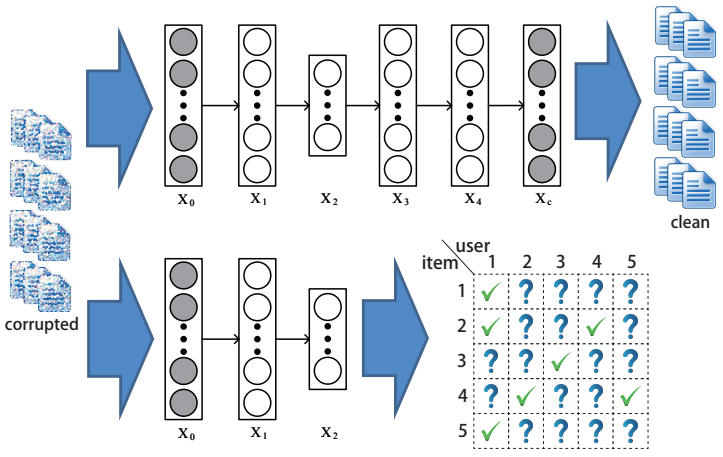

Figure 3: NN representation for degenerated CDL.

3.4 Maximum A Posteriori Estimates

Based on the CDL model above, all parameters could be treated as random variables so that fully Bayesian methods such as Markov chain Monte Carlo (MCMC) or variational approximation methods [14] may be applied. However, such treatment typically incurs high computational cost. Besides, since CTR is our primary baseline for comparison, it would be fair and reasonable to take an approach analogous to that used in CTR. Consequently, we devise below an EM-style algorithm for obtaining the MAP estimates, as in [34].

Like in CTR, maximizing the posterior probability is equivalent to maximizing the joint log-likelihood of $\mathbf{U}$, $\mathbf{V}$, $\{\mathbf{X}_l\}$, $\mathbf{X}_c$, $\{\mathbf{W}_l\}$, $\{\mathbf{b}_l\}$, and $\mathbf{R}$ given λ_u, λ_v, λ_w, λ_s, and λ_n:

$$\begin{aligned}\mathscr{L} = &-\frac{\lambda_u}{2}\sum_i \|\mathbf{u}_i\|_2^2 - \frac{\lambda_w}{2}\sum_l (\|\mathbf{W}_l\|_F^2 + \|\mathbf{b}_l\|_2^2)\\ &-\frac{\lambda_v}{2}\sum_j \|\mathbf{v}_j - \mathbf{X}_{\frac{L}{2},j*}^T\|_2^2 - \frac{\lambda_n}{2}\sum_j \|\mathbf{X}_{L,j*} - \mathbf{X}_{c,j*}\|_2^2\\ &-\frac{\lambda_s}{2}\sum_l\sum_j \|\sigma(\mathbf{X}_{l-1,j*}\mathbf{W}_l + \mathbf{b}_l) - \mathbf{X}_{l,j*}\|_2^2\\ &-\sum_{i,j}\frac{\mathbf{C}_{ij}}{2}(\mathbf{R}_{ij} - \mathbf{u}_i^T\mathbf{v}_j)^2.\end{aligned}$$

If λ_s goes to infinity, the likelihood becomes:

$$\begin{aligned}\mathscr{L} = &-\frac{\lambda_u}{2}\sum_i \|\mathbf{u}_i\|_2^2 - \frac{\lambda_w}{2}\sum_l (\|\mathbf{W}_l\|_F^2 + \|\mathbf{b}_l\|_2^2)\\ &-\frac{\lambda_v}{2}\sum_j \|\mathbf{v}_j - f_e(\mathbf{X}_{0,j*}, \mathbf{W}^+)^T\|_2^2\\ &-\frac{\lambda_n}{2}\sum_j \|f_r(\mathbf{X}_{0,j*}, \mathbf{W}^+) - \mathbf{X}_{c,j*}\|_2^2\\ &-\sum_{i,j}\frac{\mathbf{C}_{ij}}{2}(\mathbf{R}_{ij} - \mathbf{u}_i^T\mathbf{v}_j)^2, \qquad (2)\end{aligned}$$

where the encoder function $f_e(\cdot, \mathbf{W}^+)$ takes the corrupted content vector $\mathbf{X}_{0,j*}$ of item j as input and computes the encoding of the item, and the function $f_r(\cdot, \mathbf{W}^+)$ also takes $\mathbf{X}_{0,j*}$ as input, computes the encoding and then the reconstructed content vector of item j. For example, if the number of layers $L = 6$, $f_e(\mathbf{X}_{0,j*}, \mathbf{W}^+)$ is the output of the third layer while $f_r(\mathbf{X}_{0,j*}, \mathbf{W}^+)$ is the output of the sixth layer.

From the perspective of optimization, the third term in the objective function (2) above is equivalent to a multi-layer perceptron using the latent item vectors $\mathbf{v}_j$ as target while the fourth term is equivalent to an SDAE minimizing the reconstruction error. Seeing from the view of neural networks (NN), when λ_s approaches positive infinity, training of the probabilistic graphical model of CDL in Figure 1(left) would degenerate to simultaneously training two neural networks overlaid together with a common input layer (the corrupted input) but different output layers, as shown in Figure 3. Note that the second network is much more complex than typical neural networks due to the involvement of the rating matrix.

When the ratio λ_n/λ_v approaches positive infinity, it will degenerate to a two-step model in which the latent representation learned using SDAE is put directly into the CTR. Another extreme happens when λ_n/λ_v goes to zero where the decoder of the SDAE essentially vanishes. On the right of Figure 1 is the graphical model of the degenerated CDL when λ_n/λ_v goes to zero. As demonstrated in the experiments, the predictive performance will suffer greatly for both extreme cases.

For $\mathbf{u}_i$ and $\mathbf{v}_j$, coordinate ascent similar to [34, 13] is used. Given the current $\mathbf{W}^+$, we compute the gradients of $\mathscr{L}$ with respect to $\mathbf{u}_i$ and $\mathbf{v}_j$ and set them to zero, leading to the following update rules:

$$\begin{aligned}\mathbf{u}_i &\leftarrow (\mathbf{V}\mathbf{C}_i\mathbf{V}^T + \lambda_u\mathbf{I}_K)^{-1}\mathbf{V}\mathbf{C}_i\mathbf{R}_i\\ \mathbf{v}_j &\leftarrow (\mathbf{U}\mathbf{C}_i\mathbf{U}^T + \lambda_v\mathbf{I}_K)^{-1}(\mathbf{U}\mathbf{C}_j\mathbf{R}_j + \lambda_v f_e(\mathbf{X}_{0,j*}, \mathbf{W}^+)^T),\end{aligned}$$

where $\mathbf{U} = (\mathbf{u}_i)_{i=1}^I$, $\mathbf{V} = (\mathbf{v}_j)_{j=1}^J$, $\mathbf{C}_i = \mathrm{diag}(\mathbf{C}_{i1}, \ldots, \mathbf{C}_{iJ})$ is a diagonal matrix, $\mathbf{R}_i = (\mathbf{R}_{i1}, \ldots, \mathbf{R}_{iJ})^T$ is a column vector containing all the ratings of user i, and $\mathbf{C}_{ij}$ reflects the confidence controlled by a and b as discussed in [13].

Given $\mathbf{U}$ and $\mathbf{V}$, we can learn the weights $\mathbf{W}_l$ and biases $\mathbf{b}_l$ for each layer using the back-propagation learning algorithm. The gradients of the likelihood with respect to $\mathbf{W}_l$ and $\mathbf{b}_l$ are as follows:

$$\begin{aligned}\nabla_{\mathbf{W}_l}\mathscr{L} &= -\lambda_w\mathbf{W}_l\\ &-\lambda_v\sum_j \nabla_{\mathbf{W}_l} f_e(\mathbf{X}_{0,j*}, \mathbf{W}^+)^T(f_e(\mathbf{X}_{0,j*}, \mathbf{W}^+)^T - \mathbf{v}_j)\\ &-\lambda_n\sum_j \nabla_{\mathbf{W}_l} f_r(\mathbf{X}_{0,j*}, \mathbf{W}^+)(f_r(\mathbf{X}_{0,j*}, \mathbf{W}^+) - \mathbf{X}_{c,j*})\end{aligned}$$

$$\begin{aligned}\nabla_{\mathbf{b}_l}\mathscr{L} &= -\lambda_w\mathbf{b}_l\\ &-\lambda_v\sum_j \nabla_{\mathbf{b}_l} f_e(\mathbf{X}_{0,j*}, \mathbf{W}^+)^T(f_e(\mathbf{X}_{0,j*}, \mathbf{W}^+)^T - \mathbf{v}_j)\\ &-\lambda_n\sum_j \nabla_{\mathbf{b}_l} f_r(\mathbf{X}_{0,j*}, \mathbf{W}^+)(f_r(\mathbf{X}_{0,j*}, \mathbf{W}^+) - \mathbf{X}_{c,j*}).\end{aligned}$$

By alternating the update of $\mathbf{U}$, $\mathbf{V}$, $\mathbf{W}_l$, and $\mathbf{b}_l$, we can find a local optimum for $\mathscr{L}$. Several commonly used techniques such as using a momentum term may be used to alleviate the local optimum problem. For completeness, we also provide a sampling- based algorithm for CDL in the appendix.

3.5 Prediction

Let D be the observed test data. Similar to [34], we use the point estimates of $\mathbf{u}_i$, $\mathbf{W}^+$ and $\boldsymbol{\epsilon}_j$ to calculate the predicted rating:

$$E[\mathbf{R}_{ij}|D] \approx E[\mathbf{u}_i|D]^T(E[f_e(\mathbf{X}_{0,j*}, \mathbf{W}^+)^T|D] + E[\boldsymbol{\epsilon}_j|D]),$$

where $E[\cdot]$ denotes the expectation operation. In other words,

we approximate the predicted rating as:

$$\mathbf{R}^*_{ij} \approx (\mathbf{u}^*_j)^T(f_e(\mathbf{X}_{0,j*}, \mathbf{W}^{+*})^T + \boldsymbol{\epsilon}^*_j) = (\mathbf{u}^*_i)^T\mathbf{v}^*_j.$$

Note that for any new item j with no rating in the training data, its offset $\boldsymbol{\epsilon}^*_j$ will be $\mathbf{0}$.

4. EXPERIMENTS

Extensive experiments are conducted on three real-world datasets from different domains to demonstrate the effectiveness of our model both quantitatively and qualitatively[2].

4.1 Datasets

We use three datasets from different real-world domains, two from CiteULike[3] and one from Netflix, for our experiments. The first two datasets, from [35], were collected in different ways, specifically, with different scales and different degrees of sparsity to mimic different practical situations. The first dataset, *citeulike-a*, is mostly from [34]. The second dataset, *citeulike-t*, was collected independently of the first one. They manually selected 273 seed tags and collected all the articles with at least one of those tags. Similar to [34], users with fewer than 3 articles are not included. As a result, *citeulike-a* contains 5551 users and 16980 items. For *citeulike-t*, the numbers are 7947 and 25975. We can see that *citeulike-t* contains more users and items than *citeulike-a*. Also, *citeulike-t* is much sparser as only 0.07% of its user-item matrix entries contain ratings but *citeulike-a* has ratings in 0.22% of its user-item matrix entries.

The last dataset, *Netflix*, consists of two parts. The first part, with ratings and movie titles, is from the Netflix challenge dataset. The second part, with plots of the corresponding movies, was collected by us from IMDB [4]. Similar to [41], in order to be consistent with the implicit feedback setting of the first two datasets, we extract only positive ratings (rating 5) for training and testing. After removing users with less than 3 positive ratings and movies without plots, we have 407261 users, 9228 movies, and 15348808 ratings in the final dataset.

We follow the same procedure as that in [34] to preprocess the text information (item content) extracted from the titles and abstracts of the articles and the plots of the movies. After removing stop words, the top S discriminative words according to the tf-idf values are chosen to form the vocabulary (S is 8000, 20000, and 20000 for the three datasets).

4.2 Evaluation Scheme

For each dataset, similar to [35, 36], we randomly select P items associated with each user to form the training set and use all the rest of the dataset as the test set. To evaluate and compare the models under both sparse and dense settings, we set P to 1 and 10, respectively, in our experiments. For each value of P, we repeat the evaluation five times with different randomly selected training sets and the average performance is reported.

As in [34, 22, 35], we use recall as the performance measure because the rating information is in the form of implicit feedback [13, 23]. Specifically, a zero entry may be due to the fact that the user is not interested in the item, or that the user is not aware of its existence. As such, precision is not a suitable performance measure. Like most recommender systems, we sort the predicted ratings of the candidate items and recommend the top M items to the target user. The recall@M for each user is then defined as:

$$\text{recall@}M = \frac{\text{number of items that the user likes among the top } M}{\text{total number of items that the user likes}}.$$

The final result reported is the average recall over all users.

Another evaluation metric is the mean average precision (mAP). Exactly the same as [21], we set the cutoff point at 500 for each user.

4.3 Baselines and Experimental Settings

The models included in our comparison are listed as follows:

- **CMF**: Collective Matrix Factorization [30] is a model incorporating different sources of information by simultaneously factorizing multiple matrices. In this paper, the two factorized matrices are $\mathbf{R}$ and $\mathbf{X}_c$.
- **SVDFeature**: SVDFeature [8] is a model for feature-based collaborative filtering. In this paper we use the content information $\mathbf{X}_c$ as raw features to feed into SVDFeature.
- **DeepMusic**: DeepMusic [21] is a model for music recommendation mentioned in Section 1. We use the variant, a loosely coupled method, that achieves the best performance as our baseline.
- **CTR**: Collaborative Topic Regression [34] is a model performing topic modeling and collaborative filtering simultaneously as mentioned in the previous section.
- **CDL**: Collaborative Deep Learning is our proposed model as described above. It allows different levels of model complexity by varying the number of layers.

In the experiments, we first use a validation set to find the optimal hyperparameters for CMF, SVDFeature, CTR, and DeepMusic. For CMF, we set the regularization hyperparameters for the latent factors of different contexts to 10. After the grid search, we find that CMF performs best when the weights for the rating matrix and content matrix (BOW) are both 5 in the sparse setting. For the dense setting the weights are 8 and 2, respectively. For SVDFeature, the best performance is achieved when the regularization hyperparameters for the users and items are both 0.004 with the learning rate equal to 0.005. For DeepMusic, we find that the best performance is achieved using a CNN with two convolutional layers. We also try our best to tune the other hyperparameters. For CTR, we find that it can achieve good prediction performance when $\lambda_u = 0.1$, $\lambda_v = 10$, $a = 1$, $b = 0.01$, and $K = 50$ (note that a and b determine the confidence parameters $\mathbf{C}_{ij}$). For CDL, we directly set $a = 1$, $b = 0.01$, $K = 50$ and perform grid search on the hyperparameters λ_u, λ_v, λ_n, and λ_w. For the grid search, we split the training data and use 5-fold cross validation.

We use a masking noise with a noise level of 0.3 to get the corrupted input $\mathbf{X}_0$ from the clean input $\mathbf{X}_c$. For CDL with more than one layer of SDAE ($L > 2$), we use a dropout rate [2, 33, 11] of 0.1 to achieve adaptive regularization. In terms of network architecture, the number of hidden units K_l is set to 200 for l such that $l \neq L/2$ and $0 < l < L$. While both K_0 and K_L are equal to the number of words S in the dictionary, $K_{L/2}$ is set to K which is the number of dimensions of the

[2]Code and data are available at `www.wanghao.in`

[3]CiteULike allows users to create their own collections of articles. There are abstract, title, and tags for each article. More details about the CiteULike data can be found at `http://www.citeulike.org`.

[4]`http://www.imdb.com`

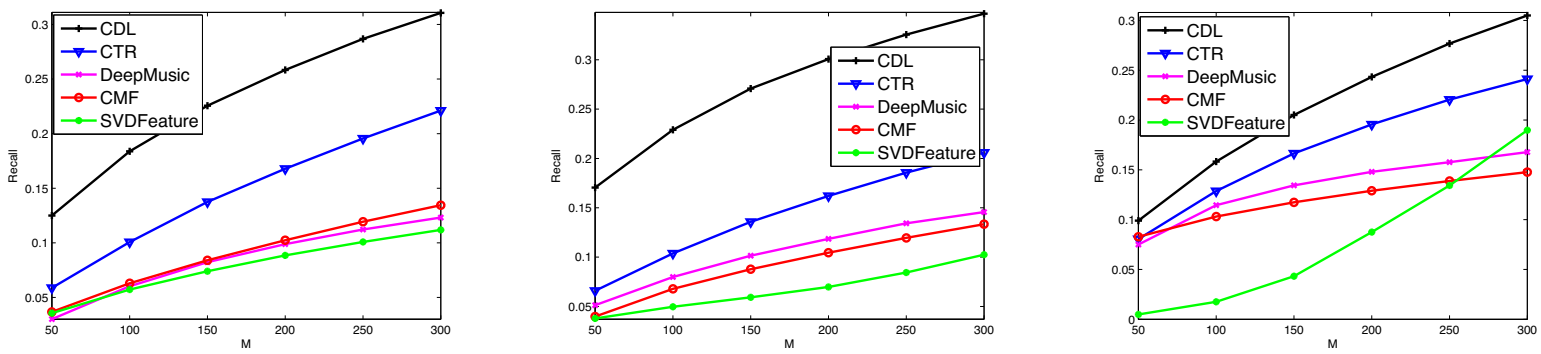

Figure 4: Performance comparison of CDL, CTR, DeepMusic, CMF, and SVDFeature based on recall@M for datasets *citeulike-a*, *citeulike-t*, and *Netflix* in the sparse setting. A 2-layer CDL is used.

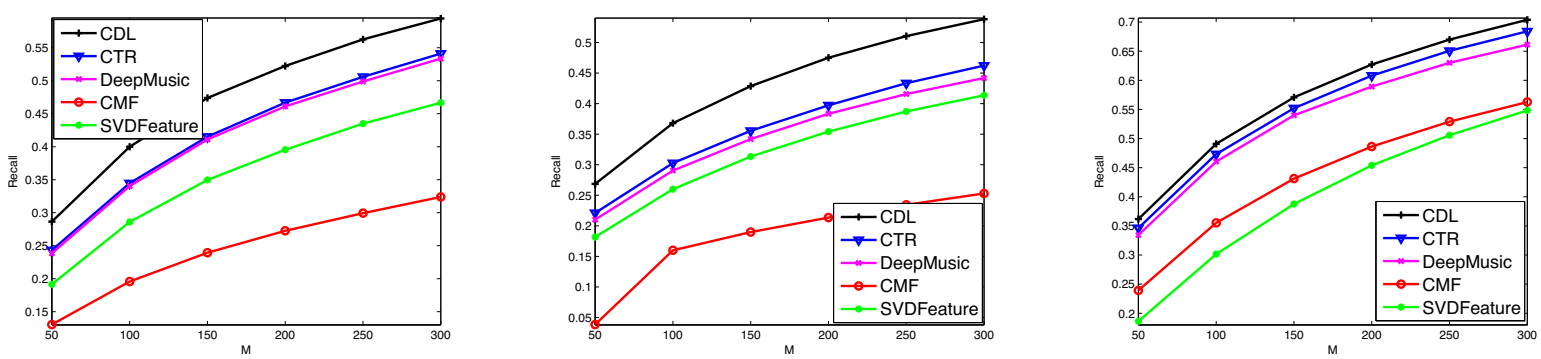

Figure 5: Performance comparison of CDL, CTR, DeepMusic, CMF, and SVDFeature based on recall@M for datasets *citeulike-a*, *citeulike-t*, and *Netflix* in the dense setting. A 2-layer CDL is used.

Table 1: mAP for three datasets

	citeulike-a	*citeulike-t*	*Netflix*
CDL	**0.0514**	**0.0453**	**0.0312**
CTR	0.0236	0.0175	0.0223
DeepMusic	0.0159	0.0118	0.0167
CMF	0.0164	0.0104	0.0158
SVDFeature	0.0152	0.0103	0.0187

Table 2: Recall@300 in the sparse setting (%)

#layers	1	2	3
citeulike-a	27.89	**31.06**	30.70
citeulike-t	32.58	34.67	**35.48**
Netflix	29.20	30.50	**31.01**

learned representation. For example, the 2-layer CDL model ($L = 4$) has a Bayesian SDAE of architecture '8000-200-50-200-8000' for the *citeulike-a* dataset.

4.4 Quantitative Comparison

Figures 4 and 5 show the results that compare CDL, CTR, DeepMusic, CMF, and SVDFeature using the three datasets under both the sparse ($P = 1$) and dense ($P = 10$) settings. We can see that CTR is a strong baseline which beats DeepMusic, CMF, and SVDFeature in all datasets even though DeepMusic has a deep architecture. In the sparse setting, CMF outperforms SVDFeature most of the time and sometimes even achieves performance comparable to CTR. DeepMusic performs poorly due to lack of ratings and overfitting. In the dense setting, SVDFeature is significantly better than CMF for *citeulike-a* and *citeulike-t* but is inferior to CMF for *Netflix*. DeepMusic is still slightly worse than CTR due to the reasons mentioned in Section 1. To focus more specifically on comparing CDL with CTR, we can see that for *citeulike-a*, 2-layer CDL outperforms CTR by a margin of 4.2%∼6.0% in the sparse setting and 3.3%∼4.6% in the dense setting. If we increase the number of layers to 3 ($L = 6$), the margin will go up to 5.8%∼8.0% and 4.3%∼5.8%, respectively. Similarly for *citeulike-t*, 2-layer CDL outperforms CTR by a margin of 10.4%∼13.1% in the sparse setting and 4.7%∼7.6% in the dense setting. When the number of layers is increased to 3, the margin will even go up to 11.0%∼14.9% and 5.2%∼8.2%, respectively. For *Netflix*, 2-layer CDL outperforms CTR by a margin of 1.9%∼5.9% in the sparse setting and 1.5%∼2.0% in the dense setting. As we can see, seamless and successful integration of deep learning and RS requires careful designs to avoid overfitting and achieve significant performance boost.

Table 1 shows the mAP for all models in the sparse settings. We can see that the mAP of CDL is almost or more than twice of CTR. Tables 2 and 3 show the recall@300 results when CDL with different numbers of layers are applied to the three datasets under both the sparse and dense settings. As we can see, for *citeulike-t* and *Netflix*, the recall increases as the number of layers increases. For *citeulike-a*, CDL starts to overfit when it exceeds two layers. Since the standard deviation is always very small ($4.31 \times 10^{-5} \sim 9.31 \times 10^{-3}$), we do not include it in the figures and tables as it is not noticeable anyway.

Note that the results are somewhat different for the first two datasets although they are from the same domain. This is due to the different ways in which the datasets were collected, as discussed above. Specifically, both the text information and the rating matrix in *citeulike-t* are much sparser.[5] By seamlessly integrating deep representation learning for content information and CF for the rating matrix, CDL can handle both the sparse rating matrix and the

[5]Each article in *citeulike-a* has 66.6 words on average and that for *citeulike-t* is 18.8.

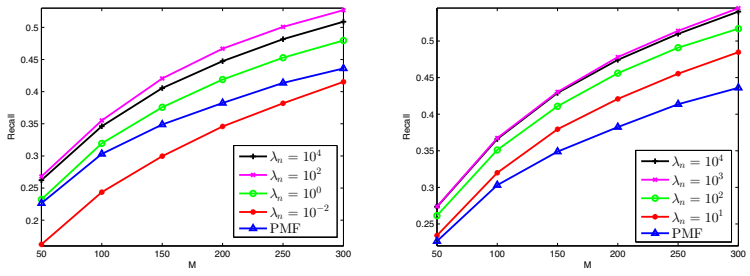

Figure 6: Performance of CDL based on recall@M for different values of λ_n on *citeulike-t*. The left plot is for $L = 2$ and the right one is for $L = 6$.

sparse text information much better and learn a much more effective latent representation for each item and hence each user.

Figure 6 shows the results for different values of λ_n using *citeulike-t* under the dense setting. We set $\lambda_u = 0.01$, $\lambda_v = 100$, and L to 2 and 6. Similar phenomena are observed when the number of layers and the value of P are varied but they are omitted here due to space constraints. As mentioned in the previous section, when λ_n is extremely large, λ_n/λ_v will approach positive infinity so that CDL degenerates to two separate models. In this case the latent item representation will be learned by the SDAE in an unsupervised manner and then it will be put directly into (a simplified version of) the CTR. Consequently, there is no interaction between the Bayesian SDAE and the collaborative filtering component based on matrix factorization and hence the prediction performance will suffer greatly. For the other extreme when λ_n is extremely small, λ_n/λ_v will approach zero so that CDL degenerates to that in Figure 1 in which the decoder of the Bayesian SDAE component essentially vanishes. This way the encoder of the Bayesian SDAE component will easily overfit the latent item vectors learned by simple matrix factorization. As we can see in Figure 6, the prediction performance degrades significantly as λ_n gets very large or very small. When $\lambda_n < 0.1$, the recall@M is already very close to (or even worse than) the result of PMF.

4.5 Qualitative Comparison

To gain a better insight into CDL, we first take a look at two example users in the *citeulike-t* dataset and represent the profile of each of them using the top three matched topics. We examine the top 10 recommended articles returned by a 3-layer ($L = 6$) CDL and CTR. The models are trained under the sparse setting ($P = 1$). From Table 4, we can speculate that user I might be a computer scientist with focus on tag recommendation, as clearly indicated by the first topic in CDL and the second one in CTR. CDL correctly recommends many articles on tagging systems while CTR focuses on social networks instead. When digging into the data, we find that the only rated article in the training data is 'What drives content tagging: the case of photos on Flickr', which is an article that talks about the impact of social networks on tagging behaviors. This may explain why CTR focuses its recommendation on social networks. On the other hand, CDL can better understand the key points of the article (i.e., tagging and CF) to make appropriate recommendation accordingly. Consequently, the precision of CDL and CTR is 70% and 10%, respectively.

From the matched topics returned by both CDL and CTR, user II might be a researcher on blood flow dynamic theory particularly in the field of medical science. CDL correctly captures the user profile and achieves a precision of 100%. However, CTR recommends quite a few articles on astronomy instead. When examining the data, we find that the only rated article returned by CTR is 'Simulating deformable particle suspensions using a coupled lattice-Boltzmann and finite-element method'. As expected, this article is on deformable particle suspension and the flow of blood cells. CTR might have misinterpreted this article, focusing its recommendation on words like 'flows' and 'formation' separately. This explains why CTR recommends articles like 'Formation versus destruction: the evolution of the star cluster population in galaxy mergers' (formation) and 'Macroscopic effects of the spectral structure in turbulent flows' (flows). As a result, its precision is only 30%.

From these two users, we can see that with a more effective representation, CDL can capture the key points of articles and the user preferences more accurately (e.g., user I). Besides, it can model the co-occurrence and relations of words better (e.g., user II).

We next present another case study which is for the *Netflix* dataset under the dense setting ($P = 10$). In this case study, we choose one user (user III) and vary the number of ratings (positive feedback) in the training set given by the user from 1 to 10. The partition of training and test data remains the same for all other users. This is to examine how the recommendation of CTR and CDL adapts as user III expresses preference for more and more movies. Table 5 shows the recommendation lists of CTR and CDL when the number of training samples is set to 2, 4, and 10. When there are only two training samples, the two movies user III likes are 'Moonstruck' and 'True Romance', which are both romance movies. For now the precision of CTR and CDL is close (20% and 30%). When two more samples are added, the precision of CDL is boosted to 50% while that of CTR remains unchanged (20%). That is because the two new movies, 'Johnny English' and 'American Beauty', belong to action and drama movies. CDL successfully captures the user's change of taste and gets two more recommendations right but CTR fails to do so. Similar phenomena can be observed when the number of training samples increases from 4 to 10. From this case study, we can see that CDL is sensitive enough to changes of user taste and hence can provide more accurate recommendation.

Table 3: Recall@300 in the dense setting (%)

#layers	1	2	3
citeulike-a	58.35	**59.43**	59.31
citeulike-t	52.68	53.81	**54.48**
Netflix	69.26	70.40	**70.42**

5. COMPLEXITY ANALYSIS AND IMPLEMENTATION

Following the update rules in this paper, the computational complexity of updating $\mathbf{u}_i$ is $O(K^2J + K^3)$, where K is the dimensionality of the learned representation and J is the number of items. The complexity for $\mathbf{v}_j$ is $O(K^2I + K^3 + SK_1)$, where I is the number of users, S is the size of the vocabulary, and K_1 is the dimensionality of the output in the first layer. Note that the third term $O(SK_1)$ is the cost of computing the output of the encoder and it is dominated by the computation of the first layer. For

Table 4: Interpretability of the latent structures learned

	user I (CDL)	in user's lib?
top 3 topics	1. search, image, query, images, queries, tagging, index, tags, searching, tag 2. social, online, internet, communities, sharing, networking, facebook, friends, ties, participation 3. collaborative, optimization, filtering, recommendation, contextual, planning, items, preferences	
top 10 articles	**1. The structure of collaborative tagging Systems**	**yes**
	2. Usage patterns of collaborative tagging systems	**yes**
	3. Folksonomy as a complex network	no
	4. HT06, tagging paper, taxonomy, Flickr, academic article, to read	**yes**
	5. Why do tagging systems work	**yes**
	6. Information retrieval in folksonomies: search and ranking	no
	7. tagging, communities, vocabulary, evolution	**yes**
	8. The complex dynamics of collaborative tagging	**yes**
	9. Improved annotation of the blogosphere via autotagging and hierarchical clustering	no
	10. Collaborative tagging as a tripartite network	**yes**
	user I (CTR)	in user's lib?
top 3 topics	1. social, online, internet, communities, sharing, networking, facebook, friends, ties, participation 2. search, image, query, images, queries, tagging, index, tags, searching, tag 3. feedback, event, transformation, wikipedia, indicators, vitamin, log, indirect, taxonomy	
top 10 articles	**1. HT06, tagging paper, taxonomy, Flickr, academic article, to read**	**yes**
	2. Structure and evolution of online social networks	no
	3. Group formation in large social networks: membership, growth, and evolution	no
	4. Measurement and analysis of online social networks	no
	5. A face(book) in the crowd: social searching vs. social browsing	no
	6. The strength of weak ties	no
	7. Flickr tag recommendation based on collective knowledge	no
	8. The computer-mediated communication network	no
	9. Social capital, self-esteem, and use of online social network sites: A longitudinal analysis	no
	10. Increasing participation in online communities: A framework for human-computer interaction	no
	user II (CDL)	in user's lib?
top 3 topics	1. flow, cloud, codes, matter, boundary, lattice, particles, galaxies, fluid, galaxy 2. mobile, membrane, wireless, sensor, mobility, lipid, traffic, infrastructure, monitoring, ad 3. hybrid, orientation, stress, fluctuations, load, temperature, centrality, mechanical, two-dimensional, heat	
top 10 articles	**1. Modeling the flow of dense suspensions of deformable particles in three dimensions**	**yes**
	2. Simplified particulate model for coarse-grained hemodynamics simulations	**yes**
	3. Lattice Boltzmann simulations of blood flow: non-newtonian rheology and clotting processes	**yes**
	4. A genome-wide association study for celiac disease identifies risk variants	**yes**
	5. Efficient and accurate simulations of deformable particles	**yes**
	6. A multiscale model of thrombus development	**yes**
	7. Multiphase hemodynamic simulation of pulsatile flow in a coronary artery	**yes**
	8. Lattice Boltzmann modeling of thrombosis in giant aneurysms	**yes**
	9. A lattice Boltzmann simulation of clotting in stented aneursysms	**yes**
	10. Predicting dynamics and rheology of blood flow	**yes**
	user II (CTR)	in user's lib?
top 3 topics	1. flow, cloud, codes, matter, boundary, lattice, particles, galaxies, fluid, galaxy 2. transition, equations, dynamical, discrete, equation, dimensions, chaos, transitions, living, trust 3. mobile, membrane, wireless, sensor, mobility, lipid, traffic, infrastructure, monitoring, ad	
top 10 articles	**1. Multiphase hemodynamic simulation of pulsatile flow in a coronary artery**	**yes**
	2. The metallicity evolution of star-forming galaxies from redshift 0 to 3	no
	3. Formation versus destruction: the evolution of the star cluster population in galaxy mergers	no
	4. Clearing the gas from globular clusters	no
	5. Macroscopic effects of the spectral structure in turbulent flows	no
	6. The WiggleZ dark energy survey	no
	7. Lattice-Boltzmann simulation of blood flow in digitized vessel networks	no
	8. Global properties of 'ordinary' early-type galaxies	no
	9. Proteus : a direct forcing method in the simulations of particulate flows	**yes**
	10. Analysis of mechanisms for platelet near-wall excess under arterial blood flow conditions	**yes**

the update of all the weights and biases, the complexity is $O(JSK_1)$ since the computation is dominated by the first layer. Thus for a complete epoch the total time complexity is $O(JSK_1 + K^2J^2 + K^2I^2 + K^3)$.

All our experiments are conducted on servers with 2 Intel E5-2650 CPUs and 4 NVIDIA Tesla M2090 GPUs each. Using the MATLAB implementation with GPU/C++ acceleration, each epoch takes only about 40 seconds and each run takes 200 epochs for the first two datasets. For *Netflix* it takes about 60 seconds per epoch and needs much fewer epochs (about 100) to get satisfactory recommendation performance. Since *Netflix* is much larger than the other two datasets, this shows that CDL is very scalable. We expect that changing the implementation to a pure C++/CUDA one would significantly reduce the time cost.

6. CONCLUSION AND FUTURE WORK

We have demonstrated in this paper that state-of-the-art performance can be achieved by jointly performing deep representation learning for the content information and collaborative filtering for the ratings (feedback) matrix. As far as we know, CDL is the first hierarchical Bayesian model to bridge the gap between state-of-the-art deep learning models and RS. In terms of learning, besides the algorithm for attaining the MAP estimates, we also derive a sampling-based algorithm for the Bayesian treatment of CDL as a Bayesian generalized version of back-propagation.

Among the possible extensions that could be made to CDL, the bag-of-words representation may be replaced by more powerful alternatives, such as [20]. The Bayesian nature of CDL also provides potential performance boost if other side information is incorporated as in [37]. Besides, as

Table 5: Example user with recommended movies

User III	Movies in the training set: **Moonstruck, True Romance, Johnny English, American Beauty, The Princess Bride, Top Gun, Double Platinum, Rising Sun, Dead Poets Society, Waiting for Guffman**		
# training samples	2	4	10
Top 10 recommended movies by CTR	Swordfish	**Pulp Fiction**	**Best in Snow**
	A Fish Called Wanda	A Clockwork Orange	**Chocolat**
	Terminator 2	Being John Malkovich	**Good Will Hunting**
	A Clockwork Orange	**Raising Arizona**	**Monty Python and the Holy Grail**
	Sling Blade	Sling Blade	Being John Malkovich
	Bridget Jones's Diary	Swordfish	**Raising Arizona**
	Raising Arizona	A Fish Called Wanda	The Graduate
	A Streetcar Named Desire	Saving Grace	Swordfish
	The Untouchables	The Graduate	Tootsie
	The Full Monty	Monster's Ball	Saving Private Ryan
# training samples	2	4	10
Top 10 recommended movies by CDL	Snatch	**Pulp Fiction**	**Good Will Hunting**
	The Big Lebowski	Snatch	**Best in Show**
	Pulp Fiction	**The Usual Suspect**	**The Big Lebowski**
	Kill Bill	Kill Bill	**A Few Good Men**
	Raising Arizona	Momento	**Monty Python and the Holy Grail**
	The Big Chill	**The Big Lebowski**	**Pulp Fiction**
	Tootsie	**One Flew Over the Cuckoo's Nest**	The Matrix
	Sense and Sensibility	As Good as It Gets	**Chocolat**
	Sling Blade	**Goodfellas**	**The Usual Suspect**
	Swinger	The Matrix	**CaddyShack**

remarked above, CDL actually provides a framework that can also admit deep learning models other than SDAE. One promising choice is the convolutional neural network model which, among other things, can explicitly take the context and order of words into account. Further performance boost may be possible when using such deep learning models.

7. ACKNOWLEDGMENTS

This research has been partially supported by research grant FSGRF14EG36.

8. REFERENCES

[1] D. Agarwal and B.-C. Chen. Regression-based latent factor models. In *KDD*, pages 19–28, 2009.

[2] P. Baldi and P. J. Sadowski. Understanding dropout. In *NIPS*, pages 2814–2822, 2013.

[3] Y. Bengio, L. Yao, G. Alain, and P. Vincent. Generalized denoising auto-encoders as generative models. In *NIPS*, pages 899–907, 2013.

[4] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[5] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent Dirichlet allocation. *JMLR*, 3:993–1022, 2003.

[6] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutiérrez. Recommender systems survey. *Knowledge Based Systems*, 46:109–132, 2013.

[7] M. Chen, Z. E. Xu, K. Q. Weinberger, and F. Sha. Marginalized denoising autoencoders for domain adaptation. In *ICML*, pages 767–774, 2012.

[8] T. Chen, W. Zhang, Q. Lu, K. Chen, Z. Zheng, and Y. Yu. Svdfeature: a toolkit for feature-based collaborative filtering. *JMLR*, 13:3619–3622, 2012.

[9] K. Georgiev and P. Nakov. A non-iid framework for collaborative filtering with restricted boltzmann machines. In *ICML*, pages 1148–1156, 2013.

[10] A. Graves, S. Fernández, F. J. Gomez, and J. Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *ICML*, pages 369–376, 2006.

[11] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580, 2012.

[12] L. Hu, J. Cao, G. Xu, L. Cao, Z. Gu, and C. Zhu. Personalized recommendation via cross-domain triadic factorization. In *WWW*, pages 595–606, 2013.

[13] Y. Hu, Y. Koren, and C. Volinsky. Collaborative filtering for implicit feedback datasets. In *ICDM*, pages 263–272, 2008.

[14] M. I. Jordan, Z. Ghahramani, T. Jaakkola, and L. K. Saul. An introduction to variational methods for graphical models. *Machine Learning*, 37(2):183–233, 1999.

[15] N. Kalchbrenner, E. Grefenstette, and P. Blunsom. A convolutional neural network for modelling sentences. *ACL*, pages 655–665, 2014.

[16] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In *NIPS*, pages 1106–1114, 2012.

[17] K. Lang. Newsweeder: Learning to filter netnews. In *ICML*, pages 331–339, 1995.

[18] W.-J. Li, D.-Y. Yeung, and Z. Zhang. Generalized latent factor models for social network analysis. In *IJCAI*, pages 1705–1710, 2011.

[19] D. J. C. MacKay. A practical Bayesian framework for backpropagation networks. *Neural Computation*, 4(3):448–472, 1992.

[20] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119, 2013.

[21] A. V. D. Oord, S. Dieleman, and B. Schrauwen. Deep content-based music recommendation. In *NIPS*, pages 2643–2651, 2013.

[22] S. Purushotham, Y. Liu, and C.-C. J. Kuo. Collaborative topic regression with social matrix factorization for recommendation systems. In *ICML*, pages 759–766, 2012.

[23] S. Rendle, C. Freudenthaler, Z. Gantner, and

L. Schmidt-Thieme. BPR: Bayesian personalized ranking from implicit feedback. In *UAI*, pages 452–461, 2009.

[24] T. N. Sainath, B. Kingsbury, V. Sindhwani, E. Arisoy, and B. Ramabhadran. Low-rank matrix factorization for deep neural network training with high-dimensional output targets. In *ICASSP*, pages 6655–6659, 2013.

[25] R. Salakhutdinov and G. E. Hinton. Deep Boltzmann machines. In *AISTATS*, pages 448–455, 2009.

[26] R. Salakhutdinov and G. E. Hinton. Semantic hashing. *Int. J. Approx. Reasoning*, 50(7):969–978, 2009.

[27] R. Salakhutdinov and A. Mnih. Probabilistic matrix factorization. In *NIPS*, pages 1257–1264, 2007.

[28] R. Salakhutdinov, A. Mnih, and G. E. Hinton. Restricted boltzmann machines for collaborative filtering. In *ICML*, pages 791–798, 2007.

[29] S. G. Sevil, O. Kucuktunc, P. Duygulu, and F. Can. Automatic tag expansion using visual similarity for photo sharing websites. *Multimedia Tools Appl.*, 49(1):81–99, 2010.

[30] A. P. Singh and G. J. Gordon. Relational learning via collective matrix factorization. In *KDD*, pages 650–658, 2008.

[31] R. S. Strichartz. *A Guide to Distribution Theory and Fourier Transforms*. World Scientific, 2003.

[32] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *JMLR*, 11:3371–3408, 2010.

[33] S. Wager, S. Wang, and P. Liang. Dropout training as adaptive regularization. In *NIPS*, pages 351–359, 2013.

[34] C. Wang and D. M. Blei. Collaborative topic modeling for recommending scientific articles. In *KDD*, pages 448–456, 2011.

[35] H. Wang, B. Chen, and W.-J. Li. Collaborative topic regression with social regularization for tag recommendation. In *IJCAI*, pages 2719–2725, 2013.

[36] H. Wang and W. Li. Relational collaborative topic regression for recommender systems. *TKDE*, 27(5):1343–1355, 2015.

[37] H. Wang, X. Shi, and D. Yeung. Relational stacked denoising autoencoder for tag recommendation. In *AAAI*, pages 3052–3058, 2015.

[38] N. Wang and D.-Y. Yeung. Learning a deep compact image representation for visual tracking. In *NIPS*, pages 809–817, 2013.

[39] X. Wang and Y. Wang. Improving content-based and hybrid music recommendation using deep learning. In *ACM MM*, pages 627–636, 2014.

[40] W. Zhang, H. Sun, X. Liu, and X. Guo. Temporal qos-aware web service recommendation via non-negative tensor factorization. In *WWW*, pages 585–596, 2014.

[41] K. Zhou and H. Zha. Learning binary codes for collaborative filtering. In *KDD*, pages 498–506, 2012.

APPENDIX

A. BAYESIAN TREATMENT FOR CDL

For completeness we also derive a sampling-based algorithm for the Bayesian treatment of CDL. It turns out to be a Bayesian and generalized version of the well-known back-propagation (BP) learning algorithm. Due to space constraints we only list the results here without detailed derivation.

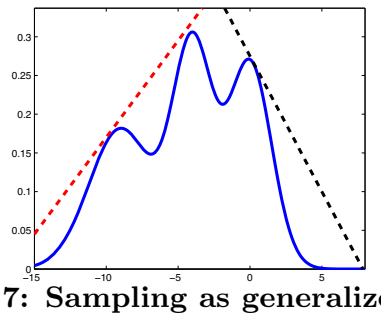

Figure 7: Sampling as generalized BP.

For $\mathbf{W}+$: We denote the concatenation of $\mathbf{W}_{l,*n}$ and $\mathbf{b}_l^{(n)}$ as $\mathbf{W}_{l,*n}^+$. Similarly, the concatenation of $\mathbf{X}_{l,j*}$ and 1 is denoted as $\mathbf{X}_{l,j*}^+$. The subscripts of $\mathbf{I}$ are ignored. Then

$$p(\mathbf{W}_{l,*n}^+|\mathbf{X}_{l-1,j*},\mathbf{X}_{l,j*},\lambda_s)$$
$$\propto \mathcal{N}(\mathbf{W}_{l,*n}^+|0,\lambda_w^{-1}\mathbf{I})\mathcal{N}(\mathbf{X}_{l,*n}|\sigma(\mathbf{X}_{l-1}^+\mathbf{W}_{l,*n}^+),\lambda_s^{-1}\mathbf{I}).$$

For $\mathbf{X}_{l,j*}$ $(l \neq L/2)$: Similarly, we denote the concatenation of $\mathbf{W}_l$ and $\mathbf{b}_l$ as $\mathbf{W}_l^+$ and have

$$p(\mathbf{X}_{l,j*}|\mathbf{W}_l^+,\mathbf{W}_{l+1}^+,\mathbf{X}_{l-1,j*},\mathbf{X}_{l+1,j*}\lambda_s)$$
$$\propto \mathcal{N}(\mathbf{X}_{l,j*}|\sigma(\mathbf{X}_{l-1,j*}^+\mathbf{W}_l^+),\lambda_s^{-1}\mathbf{I})\cdot$$
$$\mathcal{N}(\mathbf{X}_{l+1,j*}|\sigma(\mathbf{X}_{l,j*}^+\mathbf{W}_{l+1}^+),\lambda_s^{-1}\mathbf{I}).$$

Note that for the last layer ($l = L$) the second Gaussian would be $\mathcal{N}(\mathbf{X}_{c,j*}|\mathbf{X}_{l,j*},\lambda_s^{-1}\mathbf{I})$ instead.

For $\mathbf{X}_{l,j*}$ $(l = L/2)$: Similarly, we have

$$p(\mathbf{X}_{l,j*}|\mathbf{W}_l^+,\mathbf{W}_{l+1}^+,\mathbf{X}_{l-1,j*},\mathbf{X}_{l+1,j*},\lambda_s,\lambda_v,\mathbf{v}_j)$$
$$\propto \mathcal{N}(\mathbf{X}_{l,j*}|\sigma(\mathbf{X}_{l-1,j*}^+\mathbf{W}_l^+),\lambda_s^{-1}\mathbf{I})\cdot$$
$$\mathcal{N}(\mathbf{X}_{l+1,j*}|\sigma(\mathbf{X}_{l,j*}^+\mathbf{W}_{l+1}^+),\lambda_s^{-1}\mathbf{I})\mathcal{N}(\mathbf{v}_j|\mathbf{X}_{l,j*},\lambda_v^{-1}\mathbf{I}).$$

For $\mathbf{v}_j$: The posterior $p(\mathbf{v}_j|\mathbf{X}_{L/2,j*},\mathbf{R}_{*j},\mathbf{C}_{*j},\lambda_v,\mathbf{U})$

$$\propto \mathcal{N}(\mathbf{v}_j|\mathbf{X}_{L/2,j*}^T,\lambda_v^{-1}\mathbf{I})\prod_i \mathcal{N}(\mathbf{R}_{ij}|\mathbf{u}_i^T\mathbf{v}_j,\mathbf{C}_{ij}^{-1}).$$

For $\mathbf{u}_i$: The posterior $p(\mathbf{u}_i|\mathbf{R}_{i*},\mathbf{V},\lambda_u,\mathbf{C}_{i*})$

$$\propto \mathcal{N}(\mathbf{u}_i|0,\lambda_u^{-1}\mathbf{I})\prod_j (\mathbf{R}_{ij}|\mathbf{u}_i^T\mathbf{v}_j|\mathbf{C}_{ij}^{-1}).$$

Interestingly, if λ_s goes to infinity and adaptive rejection Metropolis sampling (which involves using the gradients of the objective function to approximate the proposal distribution) is used, the sampling for $\mathbf{W}^+$ turns out to be a Bayesian generalized version of BP. Specifically, as Figure 7 shows, after getting the gradient of the loss function at one point (the red dashed line on the left), the next sample would be drawn in the region under that line, which is equivalent to a probabilistic version of BP. If a sample is above the curve of the loss function, a new tangent line (the black dashed line on the right) would be added to better approximate the distribution corresponding to the loss function. After that, samples would be drawn from the region under both lines. During the sampling, besides searching for local optima using the gradients (MAP), the algorithm also takes the variance into consideration. That is why we call it Bayesian generalized back-propagation.