Deep Energy Factorization Model for Demographic Prediction

CHIH-TE LAI, National Taiwan University
CHENG-TE LI, National Cheng Kung University
SHOU-DE LIN, National Taiwan University

Demographic information is important for various commercial and academic proposes, but in reality, few of these data are accessible for analysis and research. To solve this problem, several studies predict demographic attributes from users' behavioral data. However, previous works suffer from different kinds of disadvantages. Handling data sparseness and defining useful features remain especially challenge tasks. In this article, we propose a novel *Deep Energy Factorization Model* to address these two drawbacks. The model is a designed network that performs multi-label classification and feature representation. Experiments are conducted on four datasets with four evaluation metrics. The empirical results show that our Deep Energy Factorization Model significantly outperforms state-of-the-art models.

CCS Concepts: • **Information systems** → **Data mining**; *Collaborative filtering*; *Social recommendation*; • **Computing methodologies** → *Machine learning*;

Additional Key Words and Phrases: Energy factorization, neural networks, demographic prediction, privacy preserving

ACM Reference format:
Chih-Te Lai, Cheng-Te Li, and Shou-De Lin. 2020. Deep Energy Factorization Model for Demographic Prediction. *ACM Trans. Intell. Syst. Technol.* 12, 1, Article 8 (November 2020), 16 pages.
https://doi.org/10.1145/3426240

1 INTRODUCTION

Demographic attributes are vital for businesses to plan marketing strategies, conduct market analysis, and offer personalized recommendations. However, because of privacy concerns and other reasons, it is often difficult for companies to obtain users' demographic information such as age, gender, occupation, and educational background. Kobsa et al. [8] show that users are reluctant to

This work was supported by Taiwan Ministry of Science and Technology (MOST) under grants 109-2636-E-006-017 (MOST Young Scholar Fellowship), 109-2634-F-002-033, 108-2218-E-006-036, and 109-2221-E-006-173; Academia Sinica under grant AS-TP-107-M05; Microsoft Research Asia Collaborative Project Funding (2019); and the National Center for High-Performance Computing in Taiwan.
Authors' addresses: C.-T. Lai and S.-D. Lin, National Taiwan University, No. 1, Sec. 4, Roosevelt Road, Da-an District, Taipei, Taiwan; emails: b01902067@ntu.edu.tw, sdlin@csie.ntu.edu.tw; C.-T. Li, National Cheng Kung University, No. 1, University Road, East District, Tainan, Taiwan; email: chengte@mail.ncku.edu.tw.

2157-6904/2020/11-ART8 $15.00
https://doi.org/10.1145/3426240

disclose their demographic information to recommendation systems. Dong et al. [5] also point out that a large portion of mobile users do not supply their mobile service providers with any personal information.

This issue promotes studies of a popular research topic: demographic prediction. The primary objective is to infer users' demographic attributes by exploiting their behavioral data (purchase history, ratings on items, comments on news, etc.). Although it is challenging to collect demographic data, behavioral data is relatively available for companies to acquire. In this article, we focus on inferring multiple demographic attributes based on ratings in recommendation systems.

Several studies aim to predict demographic attributes by taking advantage of various behavioral data. Weinsberg et al. [22] utilize viewers' movie ratings to predict their genders. Murray and Durrell [11] infer users' profiles by their web-browsing records. Tang et al. [17] extract researchers' profiles in an academic network. Wang et al. [21] study demographic prediction in the retail scenario. These studies reveal the fact that demographic information is predictable and has a strong connection with behavioral data.

Although demographic attributes can be inferred by behavioral data, the previous studies about demographic prediction imply limitations that do not meet real-world requests. First, some works [4, 27] predict different attributes separately; however, because of the rarity of demographic attributes, it is difficult to train a model only in terms of one single attribute. Second, many studies [4–6, 10, 16] build their models relying on manually defined features. However, defining features requires professional knowledge. Thus, it may not be feasible to generalize their models to other tasks. Finally, some works [20, 21] utilize deep neural networks to deal with large-scale datasets that suffer from the data sparsity problem, but extracting useful features from the noisy data remains a difficult task for neural networks.

To alleviate these problems, we formalize demographic prediction as a Multi-Label Classification (MLC) task and propose the Deep Energy Factorization Model (DEFM) to predict demographic attributes. DEFM is a special encoder-decoder neural network. The encoder part is a fully connected neural network that aims at generating informative codes from behavioral data. The decoder part is a hybrid network with two branches. One branch is the counterpart of the encoder that reconstructs behavioral data, and the other is a variation of Restricted Boltzmann Machine, which is for MLC. Both the encoder and decoder of DEFM are updated jointly during the training process. There are three main advantages of DEFM. First, DEFM can perform representation learning and support end-to-end training. Second, DEFM reconstructs data and handles the data sparsity problem in real-world datasets. Third, DEFM predicts multiple demographic attributes simultaneously. We verify our claims on real-world datasets with various evaluation metrics. The results show that DEFM significantly outperforms several state-of-the-art models. Our contributions are as follows:

- We present a novel learning framework, DEFM, for data representation and MLC. It boosts the performance of demographic prediction empirically.
- We propose a new hybrid model that can extract informative codes from behavioral data and jointly infer demographic attributes.
- We conduct experiments on real-world datasets, and our model outperforms state-of-the-art models in terms of several evaluation metrics.

The rest of this article is organized as follows. Section 2 briefly summarizes related work. In Section 3, we introduce our problem statement and related approaches. Section 4 elaborates the details of our model. In Section 5, we provide the experimental results. Section 6 concludes the article and provides prospective future work.

2 RELATED WORK

In this section, we review related topics: demographic prediction and MLC.

2.1 Demographic Prediction

Demographic prediction is a widely studied topic in academic areas. Several pioneers predict demographics on the basis of texts with exact semantic meaning. Bhagat et al. [1] show that users' age and gender information can be inferred from their blogs. Schler et al. [16] indicate that the strong differences in writing style and content can be utilized to infer gender and age. Otterbacher [13] conducts users' reviews of items and uses logistic regression to predict gender.

Recently, due to the growth of online social networks, lots of user data offer opportunities for demographic prediction research in social network analysis. Mislove et al. [10] reveal that users with common profiles are more likely to be friends. Dong et al. [5] propose a factor graph model to represent the interactions between users' profiles and social features on mobile communication networks. Culotta et al. [4] use logistic regression to predict demographic variables of Twitter users by whom they follow.

Commercial data (e.g., ratings on items, retail records) are also related to demographic prediction. Narayanan and Shmatikov [12] present a statistical attack for privacy breaches by calculating similarities between records. Calandrino et al. [3] use transaction history to infer other customers' transactions from their temporal changes in a recommendation system. Weinsberg et al. [22] utilize classifiers to infer gender from movie ratings. Bhagat et al. [2] also suggest a possible attack based on matrix factorization in the active learning setting. Wang et al. [20, 21] propose the Structured Neural Embedding (SNE) model and multi-task representation learning in a retail scenario. Although these models address demographic prediction, extracting latent relations among commercial data and reducing the data sparsity are rarely conducted.

2.2 Multi-Label Classification

MLC is a classification task that predicts multiple labels at once for a given data point. In other words, we assume there is a fixed number of categories for each pre-defined label. Then, given an arbitrary data point, algorithms in MLC aim to define the mapping from the data point to its labels.

The approaches can be either extending the algorithms from multi-class to multi-label or resolving the multi-label problems to multi-class problems. Adapt boosting [15], decision tree [18], k-nearest neighbors [25], and neural networks [7, 24] are the algorithms that utilize the first approach. Although some recent studies [19, 23] have developed convolutional neural network, recurrent neural network, and deep autoencoder models for MLC, they focus on preserving and modeling the semantics of image data rather than behavioral data. In other words, for behavioral data, specifically user-item or user-user interactions emphasized in this work, multiple demographic labels are depicted by how users interact with items and other users. For image data, multiple labels tend to come from objects and their relations in an image. We believe that MLCs in behavioral and image data are essentially different in encoding label information. Therefore, in this work, we do not discuss and compare MLC on image data.

3 PRELIMINARIES

In this section, we present notations and techniques to further explain our models. Briefly, we give the definition of the demographic prediction problem and describe the Neural Conditional Energy Model (NCEM), which has close connections with our model. Then, we review the concept of Denoising Autoencoders (DAE).

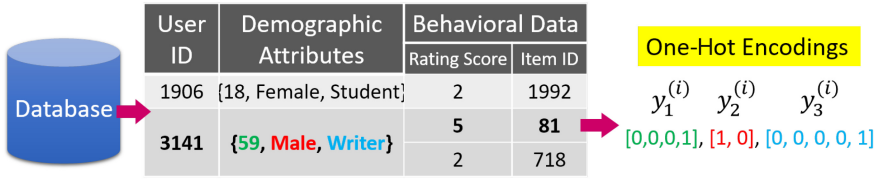


Fig. 1. A toy example for our notations.

3.1 Problem Definition

We aim to predict multiple demographic attributes based on users' ratings on items in recommendation systems. A recommender system stores users' ratings and accesses to a dataset that contains demographic attributes shared by partial users. Given users' ratings and attributes, the goal is to predict the other users' attributes.

We formulate our problem mathematically. Consider a recommendation system with ratings of M items given N users. We define r_{ij} as the rating of i-th user to j-th item, and $R \in \{1, 2, \ldots, N\} \times \{1, 2, \ldots, M\}$ denotes the set of ratings. Let $A = \{a_1, a_2, \ldots, a_K\}$ be a set of K demographic attributes. Each attribute $a_k \in A$ is associated with C_k possible values, where $C_k \geq 2, k = 1, 2, \ldots, K$. For each $i \in \{1, 2, \ldots, N\}$, we define $x^{(i)} = [r_{ij} | \forall j \in \{1, 2, \ldots, M\}, r_{ij} \in R]$ and $y^{(i)} = [y_1^{(i)}, y_2^{(i)}, \ldots, y_K^{(i)}]$, where $y_k^{(i)}$ is C_k-dimensional one-hot vector indicating the specific value of i-th users' attribute a_k, for $k = 1, 2, \ldots, K$. Let $C = \sum_k C_k$. Then, $X = [x^{(i)}]$ denotes a $(N \times M)$-dimensional matrix of ratings, $Y = [y^{(i)}]$ denotes a $(N \times C)$-dimensional matrix of one-hot encodings of demographic attributes, where $i \in \{1, 2, \ldots, N\}$. Given the preceding notations, the objective of multi-label demographic prediction is to learn a function $f : X \rightarrow Y$ such that $\hat{Y} = f(X)$, the demographic prediction, is as close to Y as possible.

Here is an example for the problem notations in Figure 1. In this example, $K = 3$, $A = \{a_1, a_2, a_3\}$, $C_1 = 4$, $C_2 = 2$, and $C_3 = 5$. a_1 denotes *age*, a_2 represents *gender*, and a_3 symbolizes *occupation*. For user $i = 3141$ and item $j = 81$, $r_{i,j} = 5$, $y^{(i)} = [y_1^{(i)}, y_2^{(i)}, y_3^{(i)}]$, where the one-hot encodings may be defined as $y_1^{(i)} = [0, 0, 0, 1]$, $y_2^{(i)} = [1, 0]$, and $y_3^{(i)} = [0, 0, 0, 0, 1]$.

3.2 Neural Conditional Energy Model

NCEM (shown in Figure 2) [7] is a hybrid model that combines the concepts of Convolutional Restricted Boltzmann Machine (CRBM) with deep neural network and is defined according to the following energy function:

$$E(v, h, y) = -f_{NN}(v; \lambda)^\top Qh - y^\top Uh - f_{NN}(v; \lambda)^\top Ly - g^\top h - s^\top y,$$

where $f_{NN}(v; \lambda)$ denotes the output given by deterministic feed-forward neural network parameterized by λ, Q is a matrix of weights between elements of $f_{NN}(v; \lambda)$ and h, U is a matrix of weights between elements of y and h, and L captures the interactions between $f_{NN}(v; \lambda)$ and y. In addition, g and s are biases for h and y, respectively. The round-rectangle nodes are embedding vectors. We have the neural network model parameters $\lambda = \{\lambda^1, \lambda^2\}$ that generate the embeddings (highlighted in blue) from the input v. Then we model the interactions between embeddings (highlighted in red) based on learnable weight matrices Q, U, and L. There is only one stochastic hidden layer, h; thus, the computation of inference is required at the last layer, which means the low-level representation can be learned efficiently using feed-forward passes.

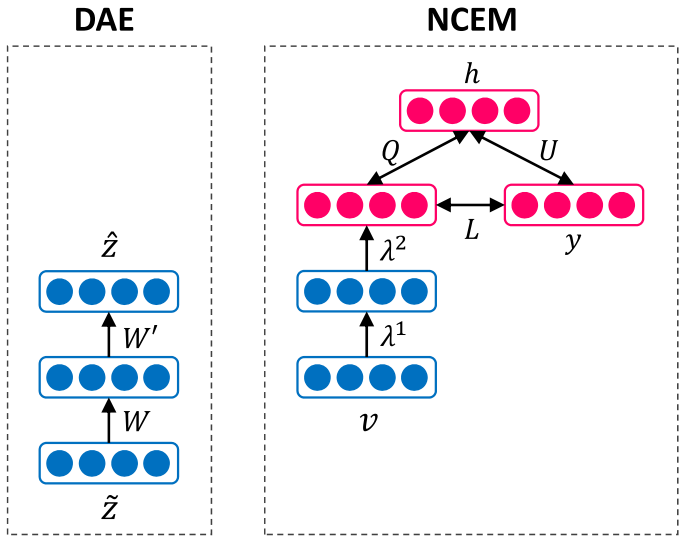


Fig. 2. Model architecture of DAE (left) and NCEM (right). Orange nodes stand for random variables, and blue nodes stand for deterministic neurons. For simplicity, the biases between layers are not shown.

To learn this model, the conditional log-likelihood is defined as

$$\frac{\partial \log p\left(y|v\right)}{\partial \lambda} = \sum_{h} p\left(h|v,y\right) \frac{\partial E\left(v,h,y\right)}{\partial \lambda} - \sum_{h,y^*} p\left(h,y^*|v\right) \frac{\partial - E\left(v,h,y^*\right)}{\partial \lambda}.$$

Considering the energy function defined previously, h denotes the stochastic hidden layer, and the deterministic hidden layers are defined in the neural network with parameter λ. Given the training set $\{V, Y\}$, one can maximize the function and learn the parameters $\{Q, U, L, g, s, \lambda\}$ with mini-batch stochastic gradient ascent. Two expectation values are required to be computed. The first term can be calculated exactly, but the second term is intractable. To infer the expectation and train NCEM, Jing and Lin [7] propose a ConditionalStochastic Back-Propagation (CSBP) algorithm based on the Contrastive Divergence (CD) method and back-propagation. Instead of starting the sampling from a random status, the CD method starts the sampling chain at a random training vector, which practically reduces plenty of time.

3.3 Denoising Autoencoder

The goal behind DAE (shown in Figure 2) is to reconstruct data from inputs of corrupted data. Since learning an identity mapping like autoencoder may not render useful features, giving the autoencoder the corrupted data is a way to avoid the issue. This approach forces the hidden layer to learn robust features.

A basic way to corrupt input data is to randomly remove some parts of the data and let the autoencoder predict these parts of data. Given input vector z, a corrupted vector z' is obtained by randomly assigning elements to zeros with some probability p, $0 < p < 1$. By the corrupted vector

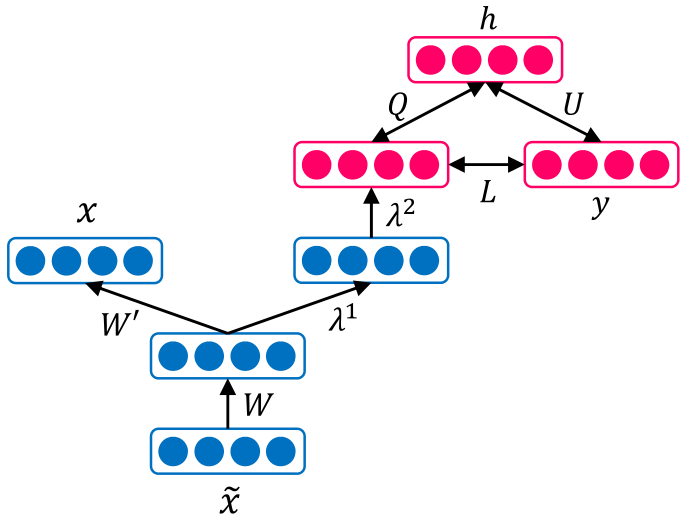


Fig. 3. Model architecture the proposed DEFM. Orange nodes stand for random variables, and blue nodes stand for deterministic neurons.

$\tilde{z}$, DAE computes a reconstruction vector $\hat{z}$.

$$z^c = f_{enc}(\tilde{z}) = \sigma(W\tilde{z} + b)$$
$$\hat{z} = f_{dec}(z^c) = \sigma(W'z^c + b')$$

DAE is trained by minimizing specific loss function l. Instead of $\tilde{z}$, the loss between z and $\hat{z}$, $l(\hat{z}, z)$, is calculated.

4 METHODOLOGY

In this section, we explain our model, DEFM, including the model architecture and the learning approach.

4.1 Deep Energy Factorization Model

DEFM is a deep neural network combined with DAE and NCEM. The overview of the proposed DEFM model is shown in Figure 3. Given the input corrupted behavioral data, we first employ DAE to remove the noise and distill useful information depicted by the latent representations. The obtained embedding vector is fed into the NCEM component in an end-to-end training manner. NCEM is devised to not only model the correlation between the feature representation and the labels but also generate the results of multi-label demographic prediction. In the model design, we have a deterministic neural network to transform the denoised embedding vector that further distills the features, as well as a stochastic hidden layer to learn how the transformed features interact with the labels. That said, the embeddings will be enhanced through modeling the deterministic and stochastic interactions between the input and labels.

DEFM first encodes the input behavioral data to latent representations, then the representations are used to reconstruct data and predict attributes. The whole model is trained and tested jointly. Specifically, given the notations in preliminaries, for each data pair $\{x, y\} \in \{X, Y\}$, the latent representation c is obtained by

$$c = f_{enc}(\tilde{x}) = W\tilde{x} + b,$$

where $\tilde{x}$ is the corrupted version of x, and the energy function of DEFM is formulated by

$$\begin{aligned} E(c,h,y) = &-f_{NN}(c;\lambda)^\top Qh - y^\top Uh \\ &- f_{NN}(c;\lambda)^\top Ly - g^\top h - s^\top y. \end{aligned}$$

Similar to NCEM, we use the following conditional probability distributions to infer y:

$$p(y|c) = \frac{\sum_h \exp(-E(c,h,y))}{\sum_{h,y'} \exp(-E(c,h,y'))}$$

$$p(h|c,y) = \prod_{j=1}^{|h|} \sigma\left(c^\top Q_{\cdot j} + y^\top U_{\cdot j} + g_j\right)$$

$$p(y|c,h) = \prod_{l=1}^{|y|} \sigma\left(c^\top L_{\cdot l} + U_{l\cdot} h + s_l\right).$$

We need to describe two noticeable things. First, the adoption of one additional fully connected layer in the encoder of DEFM comes from DAE. The idea is to derive the latent representation from the corrupted input data. Such latent representation preserves the semantics of input data and thus serves as a high-quality input of the NCEM component of our DEFM. Second, one may come up with replacing the autoencoder with a fully connected layer. However, without DAE, the sparsity and noise of the input vector cannot be addressed and alleviated. That said, we want the input vector of the NCEM component be more robust, and DAE can help achieve such a goal. Simply using a fully connected layer before NCEM is difficult to cope with corrupted, sparse, and noisy input behavioral data.

4.2 Learning

We maximize the objective function

$$\log p(y|c) - \gamma l(\hat{x}, x),$$

where $\hat{x} = f_{dec}(c)$, and γ is a regularization scalar to balance the importance of the log-likelihood $\log p(y|c)$ and the loss function $l(\hat{x}, x)$. We aim to learn parameters $\{Q, U, L, W, W', g, s, b, b', \lambda\}$ from a given training dataset $\{X, Y\}$. For an instance $\{x, y\} \in \{X, Y\}$, the gradient of the objective function with parameter θ is formulated by

$$\begin{aligned} \frac{\partial \log p(\hat{x},x)}{\partial\theta} - \gamma\frac{\partial l(\hat{x},x)}{\partial\theta} \qquad & \theta \in \{W, b\} \\ -\gamma\frac{\partial l(\hat{x},x)}{\partial\theta} \qquad & \theta \in \{W', b'\} \\ \frac{\partial \log p(y|c)}{\partial\theta} \qquad & \theta \in \{Q, U, L, g, s, \lambda\}. \end{aligned}$$

The gradient $\frac{\partial l(\hat{x},x)}{\partial\theta}$ can be calculated by using back-propagation. The gradient $\frac{\partial \log p(y|c)}{\partial\theta}$ can be written by

$$\begin{aligned} \frac{\partial \log p(y|c)}{\partial\theta} = &\sum_h p(h|c,y)\frac{\partial - E(c,h,y)}{\partial\theta} \\ &- \sum_{h,y^*} p(h,y^*|c)\frac{\partial - E(c,h,y^*)}{\partial\theta}. \end{aligned}$$

To optimize the log-likelihood, we apply CSBP [7] to estimate the intractable second term. Briefly, CSBP is an optimization algorithm combining CD method and back-propagation. The gradient of log-likelihood is approximated with CD method on sample space $\{h, y\}$, and the parameters of NN in NCEM is updated by back-propagating the gradient. We make some modification on CSBP to train DEFM. For further explanation of our learning algorithm, we rewrite the gradient $\frac{\partial \log p(y|c)}{\partial \theta}$ to free energy form

$$\frac{\partial \log p\left(y|v\right)}{\partial \lambda} = \frac{\partial - F\left(v, y\right)}{\partial \lambda} - \sum_{y^*} p\left(y^*|v\right) \frac{\partial - F\left(c, y^*\right)}{\partial \lambda},$$

where $F\left(c, y\right)$ is the free energy function that can be formulated by

$$\begin{aligned} F\left(c, y\right) &= -\log \sum_{h} \exp\left(-E\left(c, h, y\right)\right) \\ &= -\sum_{j} \log\left(1 + \exp\left(g_j + f_{NN}\left(c; \lambda\right)^\top Q_{\cdot j} + y^\top U_{\cdot j}\right)\right) \\ &\quad - f_{NN}\left(c; \lambda\right)^\top L y - s^\top y. \end{aligned}$$

We then can compute the derivatives of free energy with respect to parameters $\{Q, U, L, g, s\}$:

$$\begin{aligned} \frac{\partial F\left(c, y\right)}{\partial Q_{ij}} &= -\frac{\exp\left(g_j + f_{NN}\left(c; \lambda\right)^\top Q_{\cdot j} + y^\top U_{\cdot j}\right)}{1 + \exp\left(g_j + f_{NN}\left(c; \lambda\right)^\top Q_{\cdot j} + y^\top U_{\cdot j}\right)} c_i \\ &= -\sigma\left(g_j + f_{NN}\left(c; \lambda\right)^\top Q_{\cdot j} + y^\top U_{\cdot j}\right) c_i \\ \frac{\partial F\left(c, t\right)}{\partial U_{lj}} &= -\frac{\exp\left(g_j + f_{NN}\left(c; \lambda\right)^\top Q_{\cdot j} + y^\top U_{\cdot j}\right)}{1 + \exp\left(g_j + f_{NN}\left(c; \lambda\right)^\top Q_{\cdot j} + y^\top U_{\cdot j}\right)} y_l \\ &= -\sigma\left(g_j + f_{NN}\left(c; \lambda\right)^\top Q_{\cdot j} + y^\top U_{\cdot j}\right) y_l \\ \frac{\partial F\left(c, y\right)}{\partial L_{il}} &= -y_l f_{NN}\left(c; \lambda\right)_i \\ \frac{\partial F\left(c, y\right)}{\partial g_j} &= -\frac{\exp\left(g_j + f_{NN}\left(c; \lambda\right)^\top Q_{\cdot j} + y^\top U_{\cdot j}\right)}{1 + \exp\left(g_j + f_{NN}\left(c; \lambda\right)^\top Q_{\cdot j} + y^\top U_{\cdot j}\right)} \\ &= -\sigma\left(g_j + f_{NN}\left(c; \lambda\right)^\top Q_{\cdot j} + y^\top U_{\cdot j}\right) \\ \frac{\partial F\left(c, y\right)}{\partial s_l} &= -y_l. \end{aligned}$$

For the output of NN, we have

$$\begin{aligned} \frac{\partial F\left(c, y\right)}{\partial f_{NN}\left(c; \lambda\right)_i} &= -\sum_{j} \sigma\left(g_j + f_{NN}\left(c; \lambda\right)^\top Q_{\cdot j} + y^\top U_{\cdot j}\right) Q_{ij} \\ &\quad - \left(Ly\right)_i \end{aligned}$$

We now summarize our learning approach. First, we feed our data in DAE and pass to DNN in NCEM. Second, we use Gibbs sampling to generate the hidden layer in NCEM. Finally, we back-propagate the error signals from both NECM and DAE and update all parameters of DEFM. Formally, for a training case (x, y), our learning algorithm can be represented as the following:

(1) Given an input vector x, compute encoded vector $c = f_{enc}(\tilde{x})$ and reconstructed vector $\hat{x} = f_{dec}(c)$.

(2) Compute the loss of reconstruction $l(\hat{x}, x)$.
(3) Feed c through DNN and obtain an output $f_{NN}(c; \lambda)$.
(4) Given the output $f_{NN}(c; \lambda)$ and a label vector y, run Gibbs sampling k steps by alternately updating h and y. Get the sample y^k.
(5) For $\theta \in \{Q, U, L, W, g, s, b, \lambda\}$, update it by the following rule:

$$\theta \leftarrow \theta + \alpha \left(-\frac{\partial F(x, y)}{\partial \theta} + \frac{\partial F\left(x, y^k\right)}{\partial \theta} \right).$$

(6) For $\theta \in \{W, W', b, b'\}$, update it by the following rule:

$$\theta \leftarrow \theta - \alpha\gamma \frac{\partial l(\hat{x}, x)}{\partial \theta}$$

α: learning rate; γ: regularization weight.

4.3 Prediction

When predicting demographic attributes on a testing dataset $\{X, Y\}$, maximum *a posterior* inference is required to search for the demographic predictions $\bar{Y}$ such that the posterior probability $p(\bar{Y}|X)$ is maximized. Because maximum *a posterior* inference is intractable in DEFM, Gibbs sampling is performed to approximate the posterior probability. The prediction algorithm is similar to the one we mentioned earlier. Specifically, for one testing case (x, y), the prediction algorithm is described as follows:

(1) Compute the encoded vector $c = f_{enc}(x)$.
(2) Feed c through DNN and obtain $f_{NN}(c; \lambda)$.
(3) Generate the random hidden vector h and label vector $\hat{y}$.
(4) Given the output $f_{NN}(c; \lambda)$ and a label vector $\hat{y}$, run Gibbs sampling by k steps to alternately update h and $\hat{y}$.
(5) Use the label vector $\hat{y}$ to generate the final prediction $\bar{y}$.

Notice that we do not corrupt the input and use the reconstruction of input during the prediction phase.

5 EXPERIMENTS

We conduct experiments to illustrate the ability of demographic attributes prediction of DEFM by comparing it with baseline models on several datasets. The experimental settings are introduced first. Then for each dataset, we list tables of demographic attributes prediction evaluation results to compare DEFM with other baseline models.

5.1 Dataset

Our experiments are run on several real-world datasets, including *MovieLens 100k Dataset*, *MovieLens 1M Dataset*, *Facebook Social Network Dataset*, and *YouTube Communities Dataset*. Both *MovieLens 1M Dataset* and *MovieLens 100k Dataset* comprise exactly three demographic attributes, including age, gender, and occupation. *Facebook Social Network Dataset* and *YouTube Communities Dataset* contain users' information of social circles and communities, respectively. For research purposes, we preprocess the data from each dataset in our experiments as follows:

- *MovieLens 100K Dataset*: The *MovieLens 100K Dataset* contains 1,000 ratings from 1,000 users on 1,700 movies. We split ages to groups as 0–17, 18–35, 36–65, and 66–100. Each group is a label. We treat each occupation as a label including "retired," "none," and "other."

Table 1. Data Statistics and Density Information

	#Instances	#Interactions	Density
MovieLens100K	1,000 + 1,700	1,000	0.000588
MovieLens1M	6,000 + 4,000	1,000,000	0.041667
Facebook Social Network	4,039	88,234	0.010820
YouTube Communities	1,134,890	2,987,624	0.000005

- *MovieLens 1M Dataset*: The *MovieLens 1M Dataset* contains 1 million ratings from 6,000 users on 4,000 movies. We follow the setting of MovieLens 1M to categorize the age groups as age 0–17, age 18–24, age 25–34, age 35–44, age 45–49, and age 56–100. The occupation's labeling method is the same as the one in *MovieLens 100K Dataset*.
- *Facebook Social Network Dataset*: The *Facebook Social Network Dataset* [9] consists of 10 networks of social circles (lists of friends) from Facebook. In each network, every person has at least one friend link to another, which is recorded in its corresponding .edges file. For each person, whether she belongs to a social circle is viewed as an attribute. Each trained model aims to predict the existence in circles through the edges information for each person.
- *YouTube Communities Dataset*: The *YouTube Communities Dataset* [9] consists of one huge social network with 5,000 communities. We select the largest 10 communities and the corresponding links to serve our experiment. Given a person's links, each model aims to predict his existence in the 10 communities.

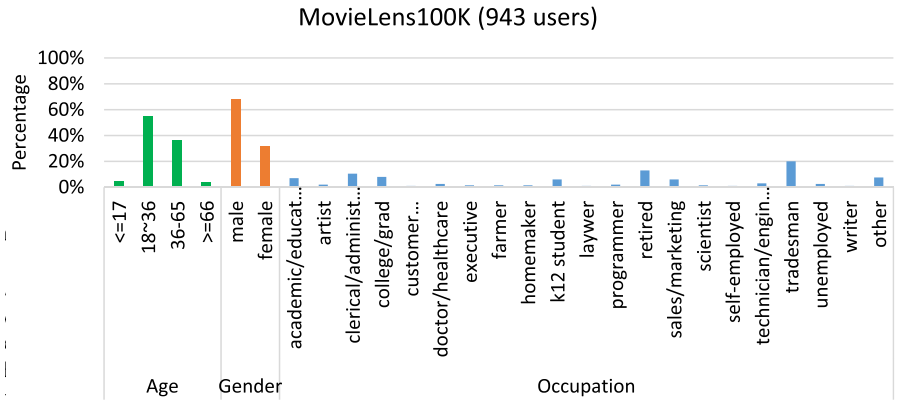
Data statistics and the attributes distributions of the MovieLens dataset are displayed in Table 1 and Figure 4, in which #Instances means the sum of the user and item numbers in MovieLens datasets and the total number of nodes in Facebook and YouTube datasets. To display how the proposed DEFM model can address the issue of data sparsity, we also calculate the density of each dataset. The density is defined as the number of interactions divided by the number of all possible interactions. It can be found that MovieLens1M and Facebook datasets have already been sparse, and MovieLens100K and YouTube datasets are extremely sparse. The experiments are conducted under such different settings of sparseness to see how DEFM and the competing methods perform.

5.2 Model Settings

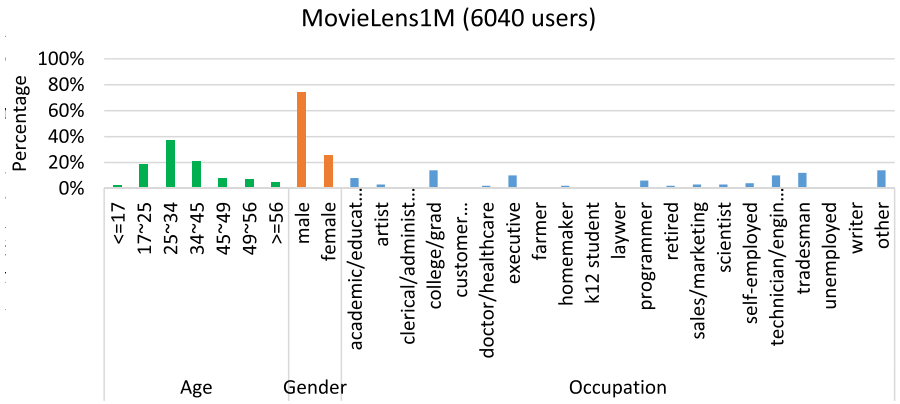
We compare DEFM with several state-of-the-art methods on demographic attributes prediction. It should be mentioned that all models in experiments have a pre-defined number of latent features. We perform the experiments with 100, 200, 300, and 400 latent features (code sizes):

- *SNE*: The negative sampling numbers is set to 1. In addition, for *MovieLens 100K Dataset* and *MovieLens 1M Dataset*, only records of ratings greater than or equivalent to 3 are used. We implemented SNE in Python at this footnote link.[1]
- *BPMF with NCEM (BFEM)*: We employ Bayesian probabilistic matrix factorization [14] to factorize the ratings and use the latent features as the input of NCEM. The hidden sizes of NCEM are set to 800, 600 for a deterministic and a stochastic layer, respectively. The negative sampling numbers, learning rate, and the iterations of sampling in training, testing phase

[1] https://github.com/LplusKira/SNE_lab.



(a) MovieLens 100K



(b) MovieLens 1M

Fig. 4. Distributions of attributes on *MovieLens 100K Dataset* and *MovieLens 1M Dataset*.

are set to 5, 0.005, 2,000, and 100. We refer the implementation of BPMF at this footnote link[2] and the implementation of NCEM at this footnote link.[3]

- *NMF with NCEM (NFEM)*: We employ non-negative matrix factorization to factorize the ratings and input use the latent features as the input of NCEM. The setting of NCEM is the same as BFEM.
- *AE with NCEM (AEEM)*: We employ DAE to learn the latent representation and use the representation as the input of NCEM. The setting of NCEM is the same as BFEM.
- *DEFM*: The hidden sizes of NCEM are set to 800, 600 for a deterministic and a stochastic layer, respectively. The regularization scalar γ between reconstruction loss and log-likelihood is 1. We use a single-layer neural network as f_{enc}. We also try f_{enc} with two layers, and it generates the same results roughly. Our source code is available at this footnote link.[4]

[2]https://www.cs.toronto.edu/~rsalakhu/BPMF.html.
[3]https://github.com/Kublai-Jing/NCEM.
[4] https://github.com/LplusKira/DEFM.

5.3 Experiment Settings

For each train data and each of the four latent features sizes, all models generate the corresponding average and standard deviation of metrics through 10-fold cross validation.

5.4 Evaluation Metrics

We consider four metrics in our experiments: Micro-F1, Ranking Loss, Average Precision, and Hamming Loss. Details can be found in the work of Zhang and Zhou [26]. Micro-F1 score and Average Precision are positive evaluations; Ranking Loss and Hamming Loss are negative evaluations. Ranking Loss and Average Precision are ranking-based measures; Micro-F1 score is a label-based measure, and Hamming Loss is an example-based measure:

- *Micro-F1 score*:
  Micro-F1 score is the harmonic mean of micro-average precision and micro-average recall. In our settings, we compute the true-positive value, false-positive value, and true-negative value for each attribute, then generate the micro-average precision and micro-average recall.
- *Ranking Loss*:
  For each prediction, Ranking Loss measures the rankings between positive and negative labels. When a negative label is predicted with a higher probability than a positive predicted, one error is counted. Ranking Loss is the average error over all combinations of positive and negative labels over all predictions.
- *Average Precision*:
  For each positive label in one prediction, two ranks of the label are considered. One is the rank of the predicted probability among all positive labels. The other is the rank of the probability among all labels. For each prediction, the rate of these two ranks is calculated. Average Precision is the average of these rates over all positive labels over all predictions.
- Hamming Loss:
  For each prediction, Hamming Loss measures the exclusive-or dissimilarity between this prediction and the real example. Hamming Loss is the average of these dissimilarities over all predictions.

5.5 Performance on Demographic Prediction

The performance of DEFM can be examined in Tables 2 and 3, which delivers two findings:

(1) DEFM has better results on *MovieLens 100k Dataset*, *MovieLens 1M Dataset*, and *YouTube Communities Dataset* in almost all metrics and code sizes. It may be due to users' attributes in training users' latent features that make DEFM generally win AEEM over MLC, considering the four metrics. In the training phase of AEEM, since the training of DAE and NCEM are performed separately, the correlation of raw behavioral data and users' attributes are not taken into account in training DAE. It turns out that by comparing DEFM with AEEM, the MLC task done by the NCEM part can be improved if users' demographic attributes are considered in training DAE. Thus, the back-propagation from the NCEM part helps generate better users' latent features. In addition, although SNE considers the correlation between users' latent features and demographic attributes in training, the generated features may not be representative enough. In our settings, SNE employs the average pooling method [21] to transform item features into each user's features. Therefore, SNE might not perform as well as our model.

(2) All models follow consistent trends on metrics for almost all datasets. For positive metrics, exactly Micro-F1 score and Average Precision, the values on any model increase as the

Table 2. Ranking Loss and Hamming Loss (mean±std) of Each Model in Datasets

Metrics	Dataset	Code	SNE	NFEM	BFEM	AEEM	DEFM
Ranking Loss	*MovieLens 100K*	100	0.192±0.002	0.169±0.012	0.180±0.019	0.152±0.007	**0.137±0.019**
		200	0.187±0.003	0.185±0.012	0.201±0.025	0.145±0.014	**0.139±0.015**
		300	0.189±0.003	0.192±0.011	0.213±0.044	0.165±0.015	**0.145±0.013**
		400	0.188±0.002	0.193±0.004	0.218±0.031	0.161±0.022	**0.144±0.006**
	MovieLens 1M	100	0.263±0.006	**0.168±0.005**	0.211±0.022	0.187±0.008	0.177±0.003
		200	0.260±0.005	0.169±0.005	0.199±0.006	0.166±0.005	**0.160±0.010**
		300	0.260±0.005	0.170±0.005	0.206±0.018	**0.155±0.008**	**0.155±0.007**
		400	0.258±0.008	0.169±0.006	0.208±0.014	0.161±0.006	**0.156±0.007**
	Facebook Social Network	100	0.382±0.034	0.023±0.008	0.081±0.022	0.026±0.010	**0.021±0.007**
		200	0.349±0.025	0.015±0.004	0.048±0.014	**0.012±0.004**	0.014±0.005
		300	0.329±0.026	**0.009±0.002**	0.030±0.009	**0.009±0.004**	0.013±0.004
		400	0.295±0.023	**0.007±0.002**	0.029±0.007	0.008±0.002	0.008±0.004
	YouTube Communities	100	0.477±0.005	0.057±0.002	0.092±0.004	0.097±0.004	**0.048±0.003**
		200	0.473±0.005	0.056±0.003	0.091±0.004	0.096±0.009	**0.046±0.002**
		300	0.469±0.007	0.058±0.002	0.096±0.009	0.088±0.009	**0.047±0.002**
		400	0.462±0.005	0.057±0.004	0.096±0.007	0.085±0.005	**0.048±0.002**
Hamming Loss	*MovieLens 100K*	100	0.481±0.005	0.092±0.012	0.100±0.010	0.091±0.010	**0.089±0.008**
		200	0.462±0.010	0.093±0.002	0.111±0.006	**0.087±0.002**	0.091±0.004
		300	0.467±0.006	0.092±0.001	0.099±0.008	**0.090±0.007**	0.092±0.006
		400	0.467±0.008	0.094±0.005	0.110±0.008	0.091±0.014	**0.088±0.003**
	MovieLens 1M	100	0.573±0.017	**0.084±0.001**	0.091±0.008	0.088±0.006	0.087±0.002
		200	0.562±0.012	**0.085±0.002**	0.093±0.007	**0.085±0.001**	**0.085±0.003**
		300	0.562±0.015	0.085±0.002	0.095±0.014	0.083±0.002	**0.082±0.002**
		400	0.557±0.019	0.086±0.001	0.092±0.007	0.083±0.002	**0.081±0.002**
	Facebook Social Network	100	0.382±0.034	**0.049±0.012**	0.118±0.020	0.056±0.015	0.051±0.011
		200	0.349±0.025	0.037±0.006	0.077±0.015	0.036±0.009	**0.035±0.007**
		300	0.329±0.026	**0.026±0.004**	0.057±0.011	0.027±0.006	0.030±0.008
		400	0.295±0.023	**0.023±0.004**	0.053±0.009	0.025±0.005	0.026±0.007
	YouTube Communities	100	0.477±0.005	0.119±0.002	0.137±0.012	0.128±0.006	**0.090±0.006**
		200	0.473±0.005	0.120±0.002	0.135±0.014	0.143±0.015	**0.083±0.002**
		300	0.469±0.007	0.120±0.001	0.135±0.013	0.144±0.016	**0.084±0.003**
		400	0.462±0.005	0.120±0.001	0.146±0.015	0.142±0.011	**0.084±0.004**

(**Bold** Indicates the Best).

code size grows up. For negative metrics, the error values decrease consistently along with the increment of code size. The main reason could be that the complexity between attributes and behavioral data cannot be depicted well without more latent features.

(3) It can be found that sometimes NFEM outperforms the proposed DEFM, especially on Facebook social network data. The social circles are considered as the demographic labels on Facebook data. A social circle is in essence a group of users who share similar attributes. A social circle is formed based on user connections in the social graph while the general demographic labels are explicitly defined and associated with users. In other words, a social circle is the product of collective behaviors from users rather than what users have by themselves. We neither design DEFM to model the collective behaviors of users nor capture the connectivity among users in DEFM. The non-negative matrix factorization

Table 3. Micro-F1 and Average Precision (Mean±std) of Each Model in Datasets

Metrics	Dataset	Code	SNE	NFEM	BFEM	AEEM	DEFM
Micro-F1 score	*MovieLens 100K*	100	0.519±0.005	0.544±0.050	0.519±0.013	0.554±0.037	**0.555±0.006**
		200	0.538±0.010	0.540±0.010	0.469±0.031	0.558±0.024	**0.566±0.014**
		300	0.533±0.006	0.543±0.017	0.492±0.036	0.541±0.026	**0.547±0.028**
		400	0.533±0.008	0.535±0.013	0.456±0.030	0.550±0.055	**0.556±0.012**
	MovieLens 1M	100	0.427±0.017	**0.476±0.014**	0.426±0.017	0.439±0.020	0.462±0.015
		200	0.438±0.012	0.482±0.015	0.427±0.016	0.473±0.012	**0.484±0.012**
		300	0.438±0.015	0.478±0.017	0.435±0.026	0.495±0.012	**0.502±0.006**
		400	0.443±0.019	0.489±0.005	0.429±0.015	0.490±0.004	**0.508±0.017**
	Facebook Social Network	100	0.618±0.034	**0.951±0.011**	0.885±0.018	0.944±0.013	0.950±0.010
		200	0.651±0.025	0.962±0.005	0.924±0.015	**0.964±0.009**	**0.964±0.007**
		300	0.671±0.026	**0.974±0.004**	0.944±0.010	0.973±0.006	0.969±0.007
		400	0.705±0.023	**0.977±0.004**	0.947±0.009	0.975±0.005	0.974±0.006
	YouTube Communities	100	0.523±0.005	0.881±0.002	0.864±0.011	0.873±0.006	**0.911±0.005**
		200	0.527±0.005	0.880±0.003	0.865±0.014	0.860±0.011	**0.917±0.003**
		300	0.531±0.007	0.880±0.002	0.866±0.011	0.858±0.014	**0.916±0.003**
		400	0.602±0.004	0.880±0.001	0.855±0.014	0.860±0.010	**0.916±0.003**
Average Precision	*MovieLens 100K*	100	0.600±0.005	0.630±0.035	0.580±0.020	0.639±0.039	**0.646±0.024**
		200	0.614±0.009	0.630±0.007	0.547±0.036	**0.651±0.021**	0.643±0.020
		300	0.611±0.004	0.619±0.008	0.559±0.048	0.630±0.027	**0.637±0.028**
		400	0.610±0.005	0.609±0.020	0.534±0.031	0.630±0.052	**0.639±0.011**
	MovieLens 1M	100	0.470±0.017	**0.581±0.006**	0.515±0.033	0.542±0.018	0.557±0.009
		200	0.480±0.014	**0.585±0.013**	0.522±0.024	0.577±0.005	0.582±0.018
		300	0.482±0.015	0.577±0.019	0.513±0.046	0.595±0.012	**0.600±0.008**
		400	0.486±0.018	0.589±0.010	0.522±0.022	0.591±0.009	**0.606±0.015**
	Facebook Social Network	100	0.648±0.029	0.978±0.008	0.921±0.021	0.975±0.010	**0.980±0.007**
		200	0.670±0.023	0.984±0.006	0.948±0.016	**0.987±0.004**	0.985±0.006
		300	0.690±0.023	**0.990±0.003**	0.966±0.011	**0.990±0.003**	0.986±0.004
		400	0.721±0.021	**0.993±0.003**	0.968±0.009	0.992±0.002	0.991±0.004
	YouTube Communities	100	0.590±0.004	0.948±0.001	0.910±0.004	0.904±0.005	**0.957±0.003**
		200	0.593±0.004	0.949±0.003	0.910±0.004	0.904±0.010	**0.957±0.003**
		300	0.597±0.005	0.946±0.003	0.904±0.010	0.915±0.009	**0.956±0.002**
		400	0.538±0.005	0.948±0.004	0.905±0.007	0.918±0.006	**0.955±0.002**

(**Bold** Indicates the Best).

method in NFEM aims at capturing the collective patterns among users. Therefore, DEFM sometimes has worse performance than NFEM.

Here we further discuss why the performance degradation owing to the design of training the autoencoder is not a major issue in our DEFM. First, DEFM is an end-to-end model that incorporates DAE with NCEM to remove the noise from the input data and model the correlation between the denoised vector and the labels. The end-to-end setting allows the model to adaptively learn the most useful information while avoiding undesired information being utilized for the prediction. If the model input vector is pre-trained by DAE first, then fed into the NCEM component, it would be more likely to cause performance degradation, as exhibited in the performance results of AEEM (DAE with NCEM). Second, in the NCEM component of our DEFM, we have a deterministic neural network to transform the latent representation that further distills the features, as well as a sin-

gle stochastic hidden layer to capture the correlations between the transformed features and the labels. That said, the embeddings are enhanced by modeling the deterministic and stochastic interactions between the input and labels. Hence, the potential performance degradation, supposed to be happen owing to autoencoder, can be avoided.

6 CONCLUSION

This article addresses the problem of multiple demographic predictions on recommendation systems. We propose the novel DEFM model, which predicts multi-label attributes and extracts useful representations in a jointly learning framework. Experiments on real-world datasets demonstrate that DEFM outperforms other state-of-the-art models on each dataset under five evaluation metrics.

Although DEFM achieves better results in demographic prediction, it has limitations. First, the usage of memory for DEFM is proportional to the number of users, which may be unacceptable when dealing with large-scale datasets. Second, a better mechanism to automatically determine the parameters in DEFM is not available. Finally, it is obscure how to select proper behavioral data and demographic attributes. We plan to tackle these issues in our extensive work.

We also perform some potential attempts. First, we aim to jointly do the recommendation and predict demographic attributes based on DEFM. A mutually reinforced mechanism may boost the performance of both tasks. Second, we plan to extend our learning algorithm to optimize a square error function and a log-likelihood function simultaneously to allow solving the tasks about multi-task learning. Finally, we aim at integrating DEFM with DAE and Restricted Boltzmann Machine, which may help create new kinds of generative models.

REFERENCES

[1] Smriti Bhagat, Irina Rozenbaum, and Graham Cormode. 2007. Applying link-based classification to label blogs. In *Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 Workshop on Web Mining and Social Network Analysis*. ACM, New York, NY, 92–101.

[2] Smriti Bhagat, Udi Weinsberg, Stratis Ioannidis, and Nina Taft. 2014. Recommending with an agenda: Active learning of private attributes using matrix factorization. In *Proceedings of the 8th ACM Conference on Recommender Systems*. ACM, New York, NY, 65–72.

[3] Joseph A. Calandrino, Ann Kilzer, Arvind Narayanan, Edward W. Felten, and Vitaly Shmatikov. 2011. "You might also like": Privacy risks of collaborative filtering. In *Proceedings of the 2011 IEEE Symposium on Security and Privacy (SP'11)*. IEEE, Los Alamitos, CA, 231–246.

[4] Aron Culotta, Nirmal Ravi Kumar, and Jennifer Cutler. 2015. Predicting the demographics of Twitter users from Website traffic data. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI'15)*. 72–78.

[5] Yuxiao Dong, Yang Yang, Jie Tang, Yang Yang, and Nitesh V. Chawla. 2014. Inferring user demographics and social strategies in mobile social networks. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, New York, NY, 15–24.

[6] Jian Hu, Hua-Jun Zeng, Hua Li, Cheng Niu, and Zheng Chen. 2007. Demographic prediction based on user's browsing behavior. In *Proceedings of the 16th International Conference on World Wide Web*. ACM, New York, NY, 151–160.

[7] How Jing and Shou-De Lin. 2014. Neural conditional energy models for multi-label classification. In *Proceedings of the 2014 IEEE International Conference on Data Mining (ICDM'14)*. IEEE, Los Alamitos, CA, 240–249.

[8] Alfred Kobsa, Bart P. Knijnenburg, and Benjamin Livshits. 2014. Let's do it at my place instead? Attitudinal and behavioral study of privacy in client-side personalization. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, New York, NY, 81–90.

[9] Jure Leskovec and Andrej Krevl. 2014. SNAP Datasets: Stanford Large Network Dataset Collection. Retrieved October 25, 2020 from http://snap.stanford.edu/data.

[10] Alan Mislove, Bimal Viswanath, Krishna P. Gummadi, and Peter Druschel. 2010. You are who you know: Inferring user profiles in online social networks. In *Proceedings of the 3rd ACM International Conference on Web Search and Data Mining*. ACM, New York, NY, 251–260.

[11] Dan Murray and Kevan Durrell. 2000. Inferring demographic attributes of anonymous Internet users. In *Web Usage Analysis and User Profiling*. Lecture Notes in Computer Science, Vol. 1836. Springer, 7–20.

[12] Arvind Narayanan and Vitaly Shmatikov. 2008. Robust de-anonymization of large sparse datasets. In *Proceedings of the IEEE Symposium on Security and Privacy (SP'08)*. IEEE, Los Alamitos, CA, 111–125.

[13] Jahna Otterbacher. 2010. Inferring gender of movie reviewers: Exploiting writing style, content and metadata. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*. ACM, New York, NY, 369–378.

[14] Ruslan Salakhutdinov and Andriy Mnih. 2008. Bayesian probabilistic matrix factorization using Markov chain Monte Carlo. In *Proceedings of the 25th International Conference on Machine Learning*. ACM, New York, NY, 880–887.

[15] Robert E. Schapire and Yoram Singer. 2000. BoosTexter: A boosting-based system for text categorization. *Machine Learning* 39, 2–3 (2000), 135–168.

[16] Jonathan Schler, Moshe Koppel, Shlomo Argamon, and James W. Pennebaker. 2006. Effects of age and gender on blogging. In *Proceedings of the AAAI Spring Symposium: Computational Approaches to Analyzing Weblogs*, Vol. 6. 199–205.

[17] Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. 2008. ArnetMiner: Extraction and mining of academic social networks. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, New York, NY, 990–998.

[18] David Vickrey, Cliff C. Lin, and Daphne Koller. 2010. Non-local contrastive objectives. In *Proceedings of the 27th International Conference on Machine Learning (ICML'10)*. 1103–1110.

[19] J. Wang, Y. Yang, J. Mao, Z. Huang, C. Huang, and W. Xu. 2016. CNN-RNN: A unified framework for multi-label image classification. In *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'16)*. 2285–2294.

[20] Pengfei Wang, Jiafeng Guo, Yanyan Lan, Jun Xu, and Xueqi Cheng. 2016. Multi-task representation learning for demographic prediction. In *Proceedings of the European Conference on Information Retrieval*. 88–99.

[21] Pengfei Wang, Jiafeng Guo, Yanyan Lan, Jun Xu, and Xueqi Cheng. 2016. Your cart tells you: Inferring demographic attributes from purchase data. In *Proceedings of the 9th ACM International Conference on Web Search and Data Mining*. ACM, New York, NY, 173–182.

[22] Udi Weinsberg, Smriti Bhagat, Stratis Ioannidis, and Nina Taft. 2012. BlurMe: Inferring and obfuscating user gender based on ratings. In *Proceedings of the 6th ACM Conference on Recommender Systems*. ACM, New York, NY, 195–202.

[23] Chih-Kuan Yeh, Wei-Chieh Wu, Wei-Jen Ko, and Yu-Chiang Frank Wang. 2017. Learning deep latent spaces for multi-label classification. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence (AAAI'17)*. 2838–2844.

[24] Min-Ling Zhang and Zhi-Hua Zhou. 2006. Multilabel neural networks with applications to functional genomics and text categorization. *IEEE Transactions on Knowledge and Data Engineering* 18, 10 (2006), 1338–1351.

[25] Min-Ling Zhang and Zhi-Hua Zhou. 2007. ML-KNN: A lazy learning approach to multi-label learning. *Pattern Recognition* 40, 7 (2007), 2038–2048.

[26] Min-Ling Zhang and Zhi-Hua Zhou. 2014. A review on multi-label learning algorithms. *IEEE Transactions on Knowledge and Data Engineering* 26, 8 (2014), 1819–1837.

[27] Yuan Zhong, Nicholas Jing Yuan, Wen Zhong, Fuzheng Zhang, and Xing Xie. 2015. You are where you go: Inferring demographic attributes from location check-ins. In *Proceedings of the 8th ACM International Conference on Web Search and Data Mining*. ACM, New York, NY, 295–304.

Received May 2019; revised July 2020; accepted September 2020