Feature Evolution Based Multi-Task Learning for Collaborative Filtering with Social Trust

Qitian Wu[1], **Lei Jiang**[1], **Xiaofeng Gao**[1], **Xiaochun Yang**[2], **Guihai Chen**[1]
[1]Shanghai Key Laboratory of Scalable Computing and Systems,
Department of Computer Science and Engineering, Shanghai Jiao Tong University
[2]Northeastern University
{echo740, johnray}@sjtu.edu.cn, gao-xf@cs.sjtu.edu.cn,
yangxc@mail.neu.edu.cn, gchen@cs.sjtu.edu.cn,

Abstract

Social recommendation could address the data sparsity and cold-start problems for collaborative filtering by leveraging user trust relationships as auxiliary information for recommendation. However, most existing methods tend to consider the trust relationship as preference similarity in a static way and model the representations for user preference and social trust via a common feature space. In this paper, we propose TrustEV and take the view of multi-task learning to unite collaborative filtering for recommendation and network embedding for user trust. We design a special feature evolution unit that enables the embedding vectors for two tasks to exchange their features in a probabilistic manner, and further harness a meta-controller to globally explore proper settings for the feature evolution units. The training process contains two nested loops, where in the outer loop, we optimize the meta-controller by Bayesian optimization, and in the inner loop, we train the feedforward model with given feature evolution units. Experiment results show that TrustEV could make better use of social information and greatly improve recommendation MAE over state-of-the-art approaches.

1 Introduction

Collaborative filtering (CF), as a powerful method for recommender systems, aims at modeling user preference and item attribute as latent factors, and use the multiplication of the two factors to estimate the probability of new user-item interaction. The approach often suffers from data sparsity and cold-start problems due to the information poverty given only user-item interaction matrix. Then social recommendation is proposed to deal with this problem by incorporating social network information into CF. Such method is based on one common knowledge that user's decisions and behaviors are often influenced by one's trust friends [Marsden and Friedkin, 1994; Mcpherson *et al.*, 2001]. Hence, one can believe that the preferences of users linked as friends share some similarities, which paves the way for using social relationships to enhance the recommendation.

Previous researches attempt to capture the social information in recommender systems from different aspects. In classical schools of thinking, researchers leverage trust propagation approaches [Massa and Avesani, 2007] to estimate user's preference via her friends' features or adopt random walk [Jamali and Ester, 2009] on user-item bipartite graph and user social graph to predict future interactions. Such approaches are called *similarity-based method*. By contrast, in modern schools of thinking, researches use different models to merge social information into recommendation, which are called *ensemble method*. They adopt matrix factorization [Ma *et al.*, 2009; Yang *et al.*, 2013; Guo *et al.*, 2015], graph regularization [Ma *et al.*, 2011; Wang *et al.*, 2017], probabilistic model [Liang *et al.*, 2016; Wang *et al.*, 2018; Meng *et al.*, 2018], network embedding [Wen *et al.*, 2018; Liu *et al.*, 2018], and graph neural network [Ying *et al.*, 2018; Song *et al.*, 2019; Wu *et al.*, 2019] to encode the high-dimensional network information as accessible features.

However, the above-mentioned methods share some common limitations: i) assume a common feature space for user preference and social trust, and directly equate the friendship as preference similarity, ii) model the social influence as static weights or fixed constraints. Due to these limitations, the incorporation of social networks could to some extent limit the accuracy of recommendation model. Firstly, in fact, for modern social network services, the relationships between friends are complicated and dynamic, and some users who possess different interests but share a common goal or mutual dependence may also connect as friends, which violates the first assumption. Secondly, the influences from friends tend to vary with different strengths in terms of distinct contexts, instead of staying unchanged as static weights. To accommodate these two facts, the model needs to make better use of the social information to exactly compensate the CF task.

*This work was supported by the National Key R&D Program of China [2018YFB1004703]; the National Natural Science Foundation of China [61872238, 61672353]; the Shanghai Science and Technology Fund [17510740200]; the Huawei Innovation Research Program [HO2018085286]; and the State Key Laboratory of Air Traffic Management System and Technology [SKLATM20180X]; the Tencent Social Ads Rhino-Bird Focused Research Program; the Fundamental Research Funds for the Central Universities (No. N171602003). Xiaofeng Gao is the corresponding author.

In this paper, we propose TrustEV which takes the view of multi-task learning to combine network embedding (NE) for social network and CF for recommendation in an elegant way. In TrustEV, for one user, we consider two independent latent factors that encode one's preference on items and trust information from social network, respectively. Then the two factors will go through a chain of feature evolution units that are specially designed to let two input vectors share their features in terms of a certain probability distribution. The feature evolution units, inspired by the independent assortment in biogenetics, possess several advantages: 1) make the two tasks achieve partial communication and model the social influence in a stochastic manner, 2) the NE feature could incorporate social information and enhance recommendation, while the CF feature could lead the NE task to capture more effective social features, 3) the probabilistic feature sharing adds enough random noises to the training and can help each task to get out of the local optimums.

Furthermore, in order to explore a proper balance for the cross-feature operation, we harness a meta-controller to efficiently search for optimal parameters for the feature evolution units. Such meta-controller would be trained by Gaussian Process (GP) based Bayesian optimization, a popular hyperparameter searching approach in AutoML [Mockus, 1974; Wang *et al.*, 2013; Kawaguchi *et al.*, 2015]. The whole training process contains two nested loops. In the inner loop, we fix the parameters for feature evolution units and train the feedforward model for CF and NE tasks. In the outer loop, we use the recommendation loss generated by the feedforward model to construct a GP and update the feature evolution units. To verify the proposed model, we conduct experiments on three benchmark datasets for social recommendation. The results show that TrustEV could improve recommendation MAE by 1.52% over several state-of-the-art approaches. Also, ablation studies manifest the effectiveness of our feature evolution unit and meta-controller.

2 Background

In this section, we review the general collaborative filtering approach and some popular social recommendation models to well point out the limitations of prior studies and the differences of our present work.

2.1 General Collaborative Filtering

Collaborative filtering deals with user-item interactions and assumes latent factors for user preference and item attribute. Use $\mathbf{p}_u$ to denote the latent factor for user u, and $\mathbf{q}_i$ to denote the latent factor for item i. Then the predicted probability that user u will click on item i (or the predicted rating that user u will give on item i) would be $\hat{r}_{ui} = \mathbf{p}_u^T \mathbf{q}_i$. Such multiplication could quantify the similarity between user preference and item attribute. Then the loss function can be defined as

$$L = \sum_{u,i} (r_{ui} - \mathbf{p}_u^T \mathbf{q}_i)^2 + \lambda_U \sum_u \|\mathbf{p}_u\|_F + \lambda_I \sum_i \|\mathbf{q}_i\|_F,$$

where r_{ui} is the ground-truth rating value, and λ_U, λ_I are regularization weights. This approach is called latent factor (LF) model or matrix factorization.

2.2 Collaborative Filtering with Social Trust

The CF method suffers from data sparsity and cold-start problems due to the sparsity of user-item interaction matrix. To tackle this issue, social recommendation is proposed and leverages social trust information to improve the recommendation accuracy. One thinking paradigm is matrix factorization, like TrustMF [Yang *et al.*, 2013] and TrustSVD [Guo *et al.*, 2015]. In TrustSVD, the predicted rating is given by

$$\hat{r}_{ui} = \mathbf{q}_i^T (\mathbf{p}_u + |I_u|^{-\frac{1}{2}} \sum_{i \in I_u} \mathbf{y}_i + |F_u|^{-\frac{1}{2}} \sum_{v \in F_u} \mathbf{w}_v),$$

where I_u denotes the set of items rated by user u, $\mathbf{y}_i$ is the latent vector for item i, F_u denotes the set of friends for user u, and $\mathbf{w}_v$ is the user-specific latent factor for friend user v. The summation $\sum_{v \in F_u} \mathbf{q}_i^T \mathbf{w}_v$ linearly aggregate the rating by user u's friends, i.e., the influence from social networks.

Another thinking paradigm is by regularization, which constrains the preference factor of a user to be similar to those of one's friends. One early study [Ma *et al.*, 2011] proposes an individual-based regularization and add a constaint term to original loss function:

$$L = \sum_{u,i} (r_{ui} - \mathbf{p}_u^T \mathbf{q}_i)^2 + \sum_u \sum_{v \in F_u} s_{uv} \cdot \|\mathbf{p}_u - \mathbf{p}_v\|_F$$
$$+ \lambda_U \sum_u \|\mathbf{p}_u\|_F + \lambda_I \sum_i \|\mathbf{q}_i\|_F,$$

where s_{uv} denotes the similarity between user u and v. Then a recent study adopts graph regularization technique to improve such model [Wang *et al.*, 2017].

Some recent works also leverage network embedding techniques to encode the social network and achieve the state-of-the-art results [Wen *et al.*, 2018; Liu *et al.*, 2018; Wu *et al.*, 2018]. Some of them are to merge the NE loss into the CF loss, while others incorporate the representations given by NE into the predicted rating [Wu *et al.*, 2019].

Nevertheless, the above methods are limited by two major concerns: 1) assume a common feature space for CF and NE representations and directly equate the friendship as preference similarity, which may limit model expressiveness for complicated social relationships, 2) model the social influence by the forms of constant weights or fixed constraints, which ignores the variant contexts and random patterns existing in social interactions and user's decisions. Differently, in our model TrustEV, we let the two tasks, CF and NE embedding, to achieve probabilistic feature sharing via a special feature evolution unit. Such mechanism can be seen as an exact compromise between general collaborative filtering and previous social recommendation methods, overcoming their respectively drawbacks and making better use of social information to enhance the recommendation accuracy.

3 Proposed Model: TrustEV

We consider a user-item interaction matrix $\mathbf{R} = \{r_{ui}\}_{M \times N}$, where M and N are numbers of users and items, respectively. If user u has rated item i, r_{ui} would be a value ranging from 1 to 5; otherwise, $r_{ui} = 0$. Also, there is a social network

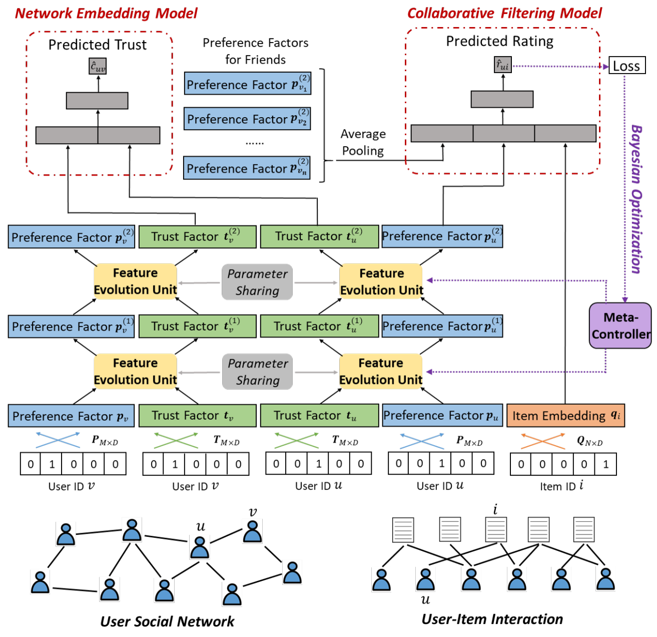


Figure 1: The framework of TrustEV model. The raw input includes user-item interactions and social network. Firstly, for user u, we consider two different embedding vectors: preference factor $\mathbf{p}_u$ and trust factor $\mathbf{t}_u$. The preference factor aims at capturing user preference on items via collaborative filtering, while the trust factor is to encode the topology information from social network. Then these two factors will go through a special feature evolution unit that lets the input vectors element-wise exchange the features in a stochastic way. The new representations will be used as input of network embedding model, which is to estimate the links between users, and collaborative filtering model, which is to predict the rating. Moreover, the above process as well as its training consists of one inner loop, and the loss generated by the current version of model would be transferred to a meta-controller that will output the updated parameter setting for the feature evolution units based on Bayesian optimization. The Bayesian optimization plays as an outer loop that aims to search for a proper balance between social network embedding and collaborative filtering

matrix $\mathbf{S} = \{s_{uv}\}_{M\times M}$, where $s_{uv} = 1$ if user u and v are friends, and $s_{uv} = 0$ otherwise. The problem is to predict future user-item interactions given some observed interactions and the social network. Fig. 1 presents the model framework of TrustEV, and we will go into the details in the following.

3.1 Embedding Layer

We encode each user as one preference factor and one trust factor. The preference factor is to capture user's interest based on user-item interaction records. We first denote user u as a one-hot vector $\mathbf{x}_u \in \mathbb{R}^{1\times M}$, where $x_{uk} = 1$ and the other elements are zeros if user u is the k-th user. Assume $\mathbf{P}_{M\times D}$ (D is the embedding size) as the embedding matrix, and the preference factor for user u would be $\mathbf{p}_u = \mathbf{P}\mathbf{x}_u^T$. Besides, the trust factor is to encode the topology information from social network. Similarly, we consider another embedding matrix $\mathbf{T}_{M\times D}$, and the trust factor for user u would be $\mathbf{t}_u = \mathbf{T}\mathbf{x}_u^T$. We also represent each item as an embedding vector to model the item attributes. For item i, we denote it as a one-hot vector $\mathbf{y}_i \in \mathbb{R}^{1\times N}$. Consider an embedding matrix $\mathbf{Q}_{N\times D}$, and the embedding factor for item i would be $\mathbf{q}_i = \mathbf{Q}\mathbf{y}_i^T$.

3.2 Collaborative Filtering and Network Embedding Models

The collaborative filtering (CF) model leverages user factor and item factor to predict the user-item interaction. Here, in order to incorporate social information, we averaged aggregate the preference factors of user's friends and obtain $\mathbf{w}_u = \frac{1}{\sqrt{|F_u|}} \sum_{v\in F_u} \mathbf{p}_v$. Then we concatenate user preference factor, item factor and friends' representations and use a two-layer neuron network to give the predicted rating value of user u on item i:

$$\hat{r}_{ui} = \mathbf{W}_2^r \tanh(\mathbf{W}_1^r[\mathbf{w}_u, \mathbf{p}_u, \mathbf{q}_i] + \mathbf{b}_1^r) + \mathbf{b}_2^r,$$

where $\mathbf{W}_1^r$, $\mathbf{W}_2^r$ are weight matrices, and $\mathbf{b}_1^r$, $\mathbf{b}_2^r$ are bias vectors.

The network embedding (NE) model aims to encode the social network topology into low-dimensional representations. To achieve this goal, we use the trust factors of users to retrieve the social network matrix $\mathbf{S}$. Also, we adopt a two-layer neuron model, which could capture enough non-linear relations, to estimate the link between user u and v:

$$\hat{s}_{uv} = \mathbf{W}_2^n \tanh(\mathbf{W}_1^n[\mathbf{t}_u, \mathbf{t}_v] + \mathbf{b}_1^n) + \mathbf{b}_2^n,$$

where $\mathbf{W}_1^n$, $\mathbf{W}_2^n$ are weight matrices, and $\mathbf{b}_1^n$, $\mathbf{b}_2^n$ are bias vectors. Then we merge the losses for two tasks as the final objective function:

$$\begin{aligned} L = &\sum_{u,i}(\hat{r}_{ui} - r_{ui})^2 + \sum_u \sum_{v\in F_u} (\hat{s}_{uv} - s_{uv})^2 \\ &+ \lambda_R(\sum_u \|\mathbf{p}_u\|_F + \sum_i \|\mathbf{q}_i\|_F) + \lambda_S \sum_u \|\mathbf{t}_u\|_F, \end{aligned} \tag{1}$$

where λ_R and λ_S are hyper-parameters that control the balance between optimization and regularization. We use stochastic gradient descendant (SGD) algorithm to optimize the objective function.

3.3 Feature Evolution Layer

To achieve enough communication between CF and NE models, we design a special *feature evolution unit*. Such unit takes the preference factor and the trust factor of a user as input, lets them randomly exchange their features element-wise, and outputs two new representations. In Fig. 1, we firstly use $\mathbf{p}_u$ and $\mathbf{t}_u$ as input, and obtain $\mathbf{p}_u^{(1)}$ and $\mathbf{t}_u^{(1)}$. Then we further get $\mathbf{p}_u^{(2)}$ and $\mathbf{t}_u^{(2)}$ based on $\mathbf{p}_u^{(1)}$ and $\mathbf{t}_u^{(1)}$. The new representations $\mathbf{p}_u^{(2)}$ and $\mathbf{t}_u^{(2)}$ would replace the original factors $\mathbf{p}_u$ and $\mathbf{t}_u$ as the input of CF and NE models.

The details for one feature evolution unit are illustrated in Fig. 2. Without loss of generality, we consider $\mathbf{x}_1$, $\mathbf{x}_2$ as the input vectors and $\mathbf{y}_1$, $\mathbf{y}_2$ as the output representations. First, we assume two Bernoulli distributions parameterized with p_1 and p_2, respectively, and construct two vectors $\mathbf{e}_1 \in \mathbb{R}^{1\times D}$, $\mathbf{e}_2 \in \mathbb{R}^{1\times D}$ whose elements are i.i.d. sampled from two Bernoulli distributions. Specifically, we have $e_{1k} \sim Bernoulli(p_1)$ (resp. $e_{2k} \sim Bernoulli(p_2)$), where

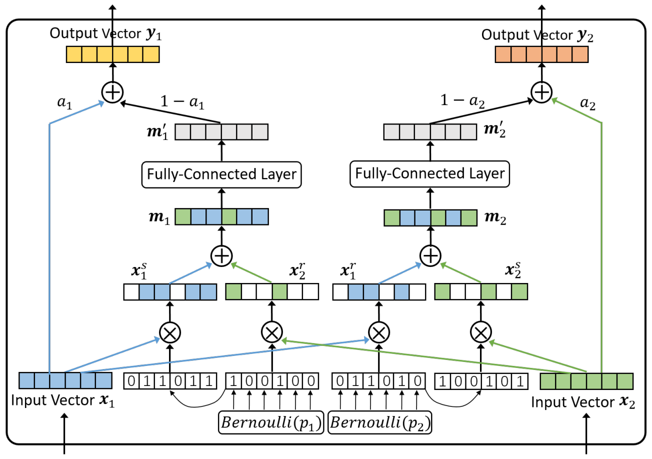


Figure 2: The details for a feature evolution unit with the input vectors $\mathbf{x}_1, \mathbf{x}_2$ and the output vectors $\mathbf{y}_1, \mathbf{y}_2$. $\otimes$ and $\oplus$ denote element-wise product and add operations, respectively

e_{1k} (resp. e_{2k}) is the k-th element of $\mathbf{e}_1$ (resp. $\mathbf{e}_2$), and $k = 1, \cdots, D$. Then we let two input vectors partially exchange their features as follows:

$$\mathbf{m}_1 = (\mathbf{1} - \mathbf{e}_1) \otimes \mathbf{x}_1 + \mathbf{e}_1 \otimes \mathbf{x}_2,$$
$$\mathbf{m}_2 = (\mathbf{1} - \mathbf{e}_2) \otimes \mathbf{x}_2 + \mathbf{e}_2 \otimes \mathbf{x}_1,$$

where $\mathbf{1} \in \mathbb{R}^{1 \times D}$ denotes a vector whose elements are all 1, and $\otimes$ denotes the element-wise product. We further let the cross-feature representations $\mathbf{m}_l$ ($l = 1, 2$) go through a fully-connected neuron layer to get a more condensed vector,

$$\mathbf{m}'_l = \tanh(\mathbf{W}_l^e \mathbf{m}_l + \mathbf{b}_l^e),$$

where $\mathbf{W}_l^e$ and $\mathbf{b}_l^e$ are weight matrix and bias vector, respectively. The output vector would be the weighted summation of the cross-feature representation and the original input vector:

$$\mathbf{y}_l = (1 - a_l) \cdot \mathbf{m}'_l + a_l \cdot \mathbf{x}_l.$$

Such mechanism is to avoid the gradient vanishing issue [He *et al.*, 2016].

The feature evolution unit is inspired by the independent assortment in biogenetics, where the descendants will inherent partial genetic traits from their parents. Once two vectors go through a feature evolution unit, the new representations would carry partial information from both vectors. To achieve a deeper feature sharing, one can concatenate K feature evolution units one behind another. The new mechanism possesses several advantages: 1) the NE feature could help to incorporate trust information into recommendation, while the CF feature could lead the NE task to select effective social information, 2) the partial communication between two tasks enable them to exactly compensate each other instead of limiting the expressiveness of the other model, 3) the probabilistic cross-feature operation adds random noises to the training and can help each task to get out of the local optimums.

3.4 Meta-Controller with Bayesian Optimization

In the feature evolution unit, there are four hyper-parameters that need to determine: p_1, p_2, a_1, a_2. They control the impact of dual cross-feature operations. One straightforward way is to manually tune these parameters based on the performance on a validation dataset. Such method would be cumbersome and could get stuck in a sub-optimal setting since the feasible region is infinite. In this paper, we leverage Bayesian optimization [Mockus, 1974; Wang *et al.*, 2013; Kawaguchi *et al.*, 2015] to automatically search for an optimal setting for the parameters in an efficient way.

The objective of parameter searching problem is to solve $\omega = \arg\max_\omega f(\omega)$, where $\omega \in \Omega$ denotes a hyper-parameter set, Ω is the feasible region and $f : \Omega \to \mathbb{R}$ is a non-convex black-box function. Here, one cannot derive the explicit form of f, and can only obtain the value of $f(\theta)$ given θ as a query. In our case, $\omega = [p_1, p_2, a_1, a_2]$, and $f(\omega) = -L(\mathcal{D}_{valid}; \theta|\omega)$, where $L(\mathcal{D}_{valid}; \theta|\omega)$ denotes the loss on validation dataset given the model parameter θ optimized by SGD conditioned on the hyper-parameter ω for feature evolution units.

We model the hyper-parameter searching as a Gaussian Process (GP)[1]. Assume a finite set of points that we have explored, the GP model is a joint Gaussian: $\mathbf{f}(\omega_{1:h}) \sim \mathcal{N}(\mathbf{m}(\omega_{1:h}), \mathbf{K})$, where $m(\cdot)$ is the mean function (usually set to be zero), $\mathbf{K}_{i,j} = \kappa(\omega_i, \omega_j)$ is the covariance, and h is the number of data points. Here, $\kappa(\cdot, \cdot)$ is a kernel function, and one common specification is the Gaussian kernel, $\kappa(x, y) = \exp(-\frac{1}{2}(x - y)^T \Sigma^{-1}(x - y))$, where Σ^{-1} is the kernel parameter matrix which can be estimated by empirical Bayesian method [Murphy, 2012]. Then, given a new data point ω_{h+1}, one can predict the value of f by the conditional distribution, conditioned on the observed entities $\mathcal{E}_h = \{\omega_{1:h}, \mathbf{f}(\omega_{1:h})\}$ and ω_{h+1}:

$$f(\omega_{h+1}) | \mathcal{E}_h, \omega_{h+1} \sim \mathcal{N}(\mu_{h+1}, \sigma^2_{h+1}),$$

where,

$$\mu_{h+1} = \kappa(\omega_{h+1}, \omega_{1:h}) \mathbf{K}^{-1} \mathbf{f}(\omega_{1:h}),$$
$$\sigma^2_{h+1} = \kappa(\omega_{h+1}, \omega_{h+1}) - \kappa(\omega_{h+1}, \omega_{1:h}) \mathbf{K}^{-1} \kappa(\omega_{1:h}, \omega_{h+1}).$$

To select the next query point, we can optimize the UCB acquisition function generated by GP:

$$\omega^*_{h+1} = \arg\max_\omega \mu(\omega|\mathcal{E}_h) + \eta \sigma(\omega|\mathcal{E}_h), \tag{2}$$

where η is a tuning parameter in the algorithm. The above objective function in our task is a convex function, so one can use a simple algorithm like gradient descendant to solve it.

We present the training algorithm for our model TrustEV with an outer meta-controller in Alg. 1.

Algorithm 1: Training Algorithm for TrustEV

```
1 for $h = 1, 2, \cdots$ do
2     Update $\theta$ by optimizing (1) given $\omega_h$;
3     Apply the new model to $\mathcal{D}_{valid}$;
4     Augment observed entities $\mathcal{E}_h = \{\mathcal{E}_{h-1}, (\omega_h, f(\omega_h))\}$;
5     Find $\omega_{h+1}$ by optimizing (2);
6 end
```

[1]The advantage of the GP prior is the simplicity of analysis and implementation. [Rasmussen and Williams, 2006]

4 Experiments

In this section, we conduct experiments on three benchmark datasets for social recommendation to verify the effectiveness of proposed model TrustEV. The statistics for three datasets are given in Table 1.

Dataset	#users	#items	#ratings	#relationships
Epinions	49,290	139,738	664,824	487,181
Ciao	7,375	99,746	280,391	111,781
FilmTrust	1,508	2,071	35,497	1,853

Table 1: Statistics of three datasets for social recommendation

We compare our method with nine baselines: PMF [Salakhutdinov and Mnih, 2007], SVD++ [Koren, 2008], NCF [He *et al.*, 2017; He *et al.*, 2018], TrustPro [Ye *et al.*, 2011], SoReg [Ma *et al.*, 2011], TrustMF [Yang *et al.*, 2013], TrustSVD [Guo *et al.*, 2015], NSCR [Wang *et al.*, 2017], SREPS [Liu *et al.*, 2018]. PMF, SVD++ and NCF are three CF methods that only leverage user-item interaction information to predict future interaction. Differently, PMF and SVD++ are two linear models, while NCF adopts neural network to capture more complicated features. NCF is a state-of-the-art CF method. The other six competitors are social recommendation models that use both user-item interaction as well as social network to conduct recommendation. TrustPro is one similarity-based model that directly uses friends' rating to predict the rating of a user. TrustMF and TrustSVD are two matrix factorization models that introduce latent factors to retrieve the social network matrix. SoReg and NSCR are regularization-based models. The difference is that SoReg is a linear model and NSCR is a deep model. SREPS is a recent work that leverages network embedding techniques to incorporate the social information. NSCR and SREPS are two powerful baselines for social recommendation.

We use mean absolute error (MAE) and root mean square error (RMSE) to evaluate the performance. The smaller value indicates the better performance for recommendation. Followed by previous work [Guo *et al.*, 2015], for each dataset, we consider two different data split ways. Firstly, we take a *All* view and randomly split the whole dataset into three parts: 70% for training, 20% for validation, and 10% for testing. Secondly, we take a *Cold-Start* view and only consider the users with less than five rated items for testing. For each model, we independently run the experiment five times and report the average results.

We implement the model by Tensorflow. Here are the specific settings for some hyper-parameters in our model: embedding size $D = 10$, regularization weights $\lambda_R = 0.0001$, $\lambda_S = 0.0001$, learning rate 0.1. We use mini-batch training to conduct model learning, and the batch size is 64. Also, for the network embedding model, we need to sample enough negative examples for training. For each link relationship, we uniformly sample $n = 5$ negative examples from all users. The parameter settings for other models are according to the reports in the previous paper.

4.1 Comparative Results

We present the comparative results on three datasets with the data split from *All* view and *Cold-Start* (CS) view in Table 2. As we can see, compared with other competitors, our model TrustEV achieves the best recommendation accuracy on three datasets, and improve the MAE by 1.63% and RMSE by 0.47% on Epinions. The results manifest that TrustEV could make better use of social information to enhance the recommendation. Besides, comparing the results of other six social recommendation models, we find that the ensemble methods outperform the similarity-based method. Also, the network embedding based model SREPS gives the best results among all baselines. That is possibly because the NE technique could capture deeper features from the social network. Furthermore, while NCF does not use any social information, its performance is still better than other simple social recommendation methods, like TrustPro, TrustMF, TrustSVD and SoReg. One possible reason is that these social recommendation models all assume fix and determined influence from user's friends, and such consideration makes the incorporation of social network limit the expressiveness of recommendation model. By constrast, in our model TrustEV, the latent factors for network embedding and recommendation tasks would partially exchange their features in a stochastic way, which enables them to exactly compensate each other.

4.2 Ablation Study

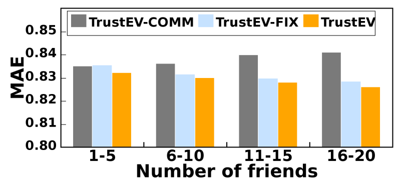
In order to verify the necessities of some key components in our model, like feature evolution units and meta-controller, we design ablation studies for TrustEV. Here we consider two simplified models: TrustEV-COMM and TrustEV-FIX. Specifically, TrustEV-FIX removes the Bayesian optimization based meta-controller, and fix the hyper-parameters for feature evolution units with $p_1 = p_2 = a_1 = a_2 = 0.5$. In, TrustEV-COMM, we set $p_1 = 1$ and $p_2 = a_1 = a_2 = 0$, in which situation, there is no cross-feature operation, and the CF and NE models share a common feature space. We show MAEs for users with different numbers of friends in Fig. 3.

As shown in Fig. 3, the performance of TrustEV-FIX is better than TrustEV-COMM, which verifies the effectiveness of feature evolution unit and shows that the cross-feature operation could indeed unite the CF and NE tasks. Moreover, we find that, as the number of friends increases, the performance of TrustEV-FIX becomes better while the performance of TrustEV-COMM becomes worse. Such phenomenon is possibly due to the fact that in TrustEV-COMM, the NE model constrains the preference factors of friend users to be similar, which to some extent limits the expressiveness of CF model (like previous social recommendation models).

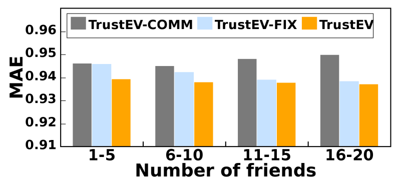
Comparing TrustEV-FIX with TrustEV, we can find that the meta-controller could indeed help to search for a proper balance of cross-feature operation and improve the performance. As the number of friends increases, the MAE of TrustEV goes down, which indicates that TrustEV is able to select more effective NE features for CF given more trust information. Also, the meta-controller requires an outer loop for global optimization of hyper-parameters, which increases the computational costs. Hence, for practical application, there would be a trade-off between accuracy and complexity.

	Epinions-*All*		Ciao-*All*		FilmTrust-*All*		Epinions-*CS*		Ciao-*CS*		FilmTrust-*CS*	
Methods	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**
PMF	0.8662	1.1089	0.9883	1.2793	0.6751	0.8112	0.9223	1.1633	0.9991	1.2894	0.7380	0.8736
SVD++	0.8482	1.0828	0.9689	1.2598	0.6419	0.8001	0.8991	1.1374	0.9795	1.2699	0.7078	0.8640
NCF	0.8411	1.0833	0.9615	1.2546	0.6338	0.7901	0.8994	1.1333	0.9716	1.2651	0.6985	0.8601
TrustPro	0.8873	1.1445	1.0089	1.3181	0.7124	0.8326	0.9463	1.1956	1.0192	1.3284	0.7788	0.8995
SocialReg	0.8499	1.0985	0.9720	1.2757	0.6317	0.8045	0.9013	1.1576	0.9828	1.2866	0.6948	0.8651
TrustMF	0.8485	1.0902	0.9696	1.2634	0.6345	0.8012	0.9075	1.1454	0.9803	1.2740	0.7020	0.8667
TrutsSVD	0.8393	1.0933	0.9614	1.2675	0.6221	0.7910	0.8938	1.1510	0.9721	1.2776	0.6904	0.8523
NSCR	0.8374	1.0881	0.9501	1.2666	0.6198	0.7834	0.8945	1.1421	0.9695	1.2775	0.6891	0.8522
SREPS	0.8284	1.0806	0.9520	1.2555	0.6182	0.7882	0.8873	1.1378	0.9605	1.2660	0.6821	0.8555
TrustEV	**0.8151**	**1.0755**	**0.9388**	**1.2491**	**0.6089**	**0.7801**	**0.8654**	**1.1319**	**0.9490**	**1.2599**	**0.6754**	**0.8405**
Impr.	1.63%	0.47%	1.21%	0.43%	1.53%	0.42%	2.53%	0.12%	1.21%	0.41%	1.00%	1.40%

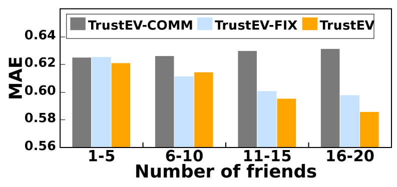
Table 2: The experiment results for proposed model TrustEV and nine competitors on three datasets with data split from two views. The orange cells mark the best values in one column, and the purple cells mark the best values among all competitors



(a) Epinions

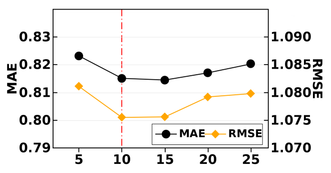


(b) Ciao

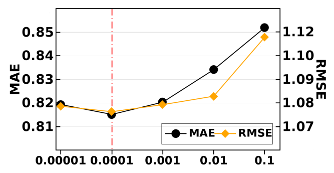


(c) FilmTrust

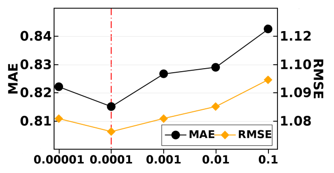
Figure 3: Ablation studies for feature evolution units and meta-controller in TrustEV on three datasets. In TrustEV-FIX, we remove the meta-controller and fix the parameter settings for feature evolution units, while in TrustEV-COMM, we remove the feature evolution units and consider one common feature space for CF and NE models



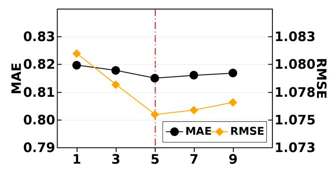
(a) D



(b) λ_R



(c) λ_S



(d) n

Figure 4: Performance variation w.r.t hyper-parameters in TrustEV, including embedding size D, regularization weights λ_R, λ_S, and negative sample size n. The vertical dotted line marks the value set in this paper

4.3 Parameter Sensitivity

We also discuss the performance variation w.r.t some model parameters, including the embedding size D, the regularization weights λ_R, λ_S, and the negative sample size n. The results are shown in Fig. 4. With the embedding size increasing from 5 to 25, the model performance firstly becomes better and then gets worse, This is because a larger embedding space could improve the model expressiveness ,but if it is too large, the representation would become sparse, which leads to performance decline. Secondly, two regularization weights have a great impact on both MAE and RMSE. Such phenomenon shows that the model requires a proper trade-off between optimization and regularization. Thirdly, we test the model with different negative sample sizes. We find that when we sample a few or a lot of negative links, the performance is not that good, and $n = 5$ is a relatively nice setting.

5 Conclusion

In this paper, we propose a new feature evolution based multi-task learning framework for social recommendation. The new model contains a special feature evolution unit that makes the latent factors for CF and NE tasks element-wise exchange their features in a random way. Such mechanism can be seen as a compromise between general CF method, where no social information is considered, and previous social recommendation methods, which imposes hard constraints on similarities of preference factors among friend users. Moreover, we design a meta-controller that is based on Bayesian optimization to automatically search for a proper hyper-parameter setting for the feature evolution unit, which controls the impact of cross-feature operation. The experiments on three benchmark datasets show that 1) the feature evolution unit helps to make better use of social information and enhance the recommendation accuracy, and 2) the meta-controller could find an optimal balance where the NE task could exactly compensate the CF. The feature evolution units can be easily generalized to other multi-task learning scenarios where two different models need to be trained and share some relations. We leave such exploration for future works.

References

[Guo *et al.*, 2015] Guibing Guo, Jie Zhang, and Neil Yorke-Smith. Trustsvd: Collaborative filtering with both the explicit and implicit influence of user trust and of item ratings. In *AAAI*, pages 123–129, 2015.

[He *et al.*, 2016] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *ECCV*, pages 630–645, 2016.

[He *et al.*, 2017] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *WWW*, pages 173–182, 2017.

[He *et al.*, 2018] Xiangnan He, Xiaoyu Du, Xiang Wang, Feng Tian, Jinhui Tang, and Tat-Seng Chua. Outer product-based neural collaborative filtering. In *IJCAI*, pages 2227–2233, 2018.

[Jamali and Ester, 2009] Mohsen Jamali and Martin Ester. Trustwalker: a random walk model for combining trust-based and item-based recommendation. In *KDD*, pages 397–406, 2009.

[Kawaguchi *et al.*, 2015] Kenji Kawaguchi, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. Bayesian optimization with exponential convergence. In *NIPS*, pages 2809–2817, 2015.

[Koren, 2008] Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *KDD*, pages 426–434, 2008.

[Liang *et al.*, 2016] Dawen Liang, Laurent Charlin, James McInerney, and David M. Blei. Modeling user exposure in recommendation. In *WWW*, pages 951–961, 2016.

[Liu *et al.*, 2018] Chun-Yi Liu, Chuan Zhou, Jia Wu, Yue Hu, and Li Guo. Social recommendation with an essential preference space. In *AAAI*, pages 346–353, 2018.

[Ma *et al.*, 2009] Hao Ma, Irwin King, and Michael R. Lyu. Learning to recommend with social trust ensemble. In *SIGIR*, pages 203–210, 2009.

[Ma *et al.*, 2011] Hao Ma, Dengyong Zhou, Chao Liu, Michael R. Lyu, and Irwin King. Recommender systems with social regularization in social networks. In *WSDM*, pages 287–296, 2011.

[Marsden and Friedkin, 1994] Peter V Marsden and Noah E Friedkin. Network studies of social influence. *Sociological Methods Research*, 22(1):127–151, 1994.

[Massa and Avesani, 2007] Paolo Massa and Paolo Avesani. Trust-aware recommender systems. In *RecSys*, pages 17–24, 2007.

[Mcpherson *et al.*, 2001] Miller Mcpherson, Lynn Smith-Lovin, and James M. Cook. Birds of a feather: Homophily in social networks. *Annual Review of Sociology*, 27(1):415–444, 2001.

[Meng *et al.*, 2018] Yitong Meng, Guangyong Chen, Jiajin Li, and Shengyu Zhang. Psrec: social recommendation with pseudo ratings. In *Proceedings of the 12th ACM Conference on Recommender Systems, RecSys 2018, Vancouver, BC, Canada, October 2-7, 2018*, pages 397–401, 2018.

[Mockus, 1974] Jonas Mockus. On bayesian methods for seeking the extremum. In *IFIP Technical Conference on Optimization Techniques*, pages 400–404, 1974.

[Murphy, 2012] Kevin P. Murphy. *Machine learning - a probabilistic perspective*. Adaptive computation and machine learning series. MIT Press, 2012.

[Rasmussen and Williams, 2006] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian processes for machine learning*. Adaptive computation and machine learning. MIT Press, 2006.

[Salakhutdinov and Mnih, 2007] Ruslan Salakhutdinov and Andriy Mnih. Probabilistic matrix factorization. In *NIPS*, pages 1257–1264, 2007.

[Song *et al.*, 2019] Weiping Song, Zhiping Xiao, Yifan Wang, Laurent Charlin, Ming Zhang, and Jian Tang. Session-based social recommendation via dynamic graph attention networks. In *WSDM*, pages 555–563, 2019.

[Wang *et al.*, 2013] Ziyu Wang, Masrour Zoghi, Frank Hutter, David Matheson, and Nando de Freitas. Bayesian optimization in high dimensions via random embeddings. In *IJCAI*, pages 1778–1784, 2013.

[Wang *et al.*, 2017] Xiang Wang, Xiangnan He, Liqiang Nie, and Tat-Seng Chua. Item silk road: Recommending items from information domains to social users. In *SIGIR*, pages 185–194, 2017.

[Wang *et al.*, 2018] Menghan Wang, Xiaolin Zheng, Yang Yang, and Kun Zhang. Collaborative filtering with social exposure: A modular approach to social recommendation. In *AAAI*, pages 2516–2523, 2018.

[Wen *et al.*, 2018] Yufei Wen, Lei Guo, Zhumin Chen, and Jun Ma. Network embedding based recommendation method in social networks. In *WWW*, pages 11–12, 2018.

[Wu *et al.*, 2018] Le Wu, Peijie Sun, Richang Hong, Yanjie Fu, Xiting Wang, and Meng Wang. Socialgcn: An efficient graph convolutional network based model for social recommendation. *CoRR*, abs/1811.02815, 2018.

[Wu *et al.*, 2019] Qitian Wu, Hengrui Zhang, Xiaofeng Gao, Peng He, Paul Weng, Han Gao, and Guihai Chen. Dual graph attention networks for deep latent representation of multifaceted social effects in recommender systems. In *WWW*, pages 2091–2102, 2019.

[Yang *et al.*, 2013] Bo Yang, Yu Lei, Dayou Liu, and Jiming Liu. Social collaborative filtering by trust. In *IJCAI*, pages 2747–2753, 2013.

[Ye *et al.*, 2011] Mao Ye, Peifeng Yin, Wang-Chien Lee, and Dik Lun Lee. Exploiting geographical influence for collaborative point-of-interest recommendation. In *SIGIR*, pages 325–334, 2011.

[Ying *et al.*, 2018] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *KDD*, pages 974–983, 2018.