A Probabilistic Rating Auto-encoder for Personalized Recommender Systems

Huizhi Liang
The University of Melbourne
VIC 3010, Melbourne
oklianghuizi@gmail.com

Timothy Baldwin
The University of Melbourne
VIC 3010, Melbourne
tb@ldwin.net

ABSTRACT

User profiling is a key component of personalized recommender systems, and is used to generate user profiles that describe individual user interests and preferences. The increasing availability of big data is driving the urgent need for user profiling algorithms that are able to generate accurate user profiles from large-scale user behavior data. In this paper, we propose a probabilistic rating auto-encoder to perform unsupervised feature learning and generate latent user feature profiles from large-scale user rating data. Based on the generated user profiles, *neighborhood* based collaborative filtering approaches have been adopted to make personalized rating predictions. The effectiveness of the proposed approach is demonstrated in experiments conducted on a real-world rating dataset from `yelp.com`.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval-Information Filtering

General Terms

Algorithms; Experimentation; Performance

Keywords

User profiling; Recommender Systems; Auto-encoder; Dimensionality Reduction

1. INTRODUCTION

Recommender systems are popular for personalization, and can help to reduce information overload for users in online communities, i.e., making suggestions regarding which information is most relevant to an individual user [1]. User profiling is the foundation of personalized recommender systems, and provides the basis to discover knowledge about users' interests, preferences, and information needs, from user-generated social data including user-generated content and behavioral information [8, 9]. With the rapid growth of social data, methods that support efficient and effective extraction of features from big social data are becoming increasingly important.

Rating (explicit or implicit) is one type of user behaviour information that has been popularly used to profile users' individual preferences and interests. For example, a user may use a rating score ranging from 1 to 5 to express their preferences or sentiment orientation with respect to an item. The typical approach is to profile users directly based on their rating behavior, and based on this, to use *neighborhood* based collaborative filtering approaches [11] to make rating predictions or top-N item recommendations [1]. Learning latent representations of users or items from user rating behavior can find latent or hidden relations between users or items, and reduce data dimensionality, thus contributing to scalable recommender systems. Although some dimensionality reduction approaches such as matrix factorization [5] have been proposed, the development of methods for learning latent representations from large-scale user rating behaviour data remains an open research question.

Deep learning techniques [4] have recently emerged as a promising framework for automatically training models over large-scale unlabeled and labeled datasets. Notably, auto-encoders [12] can learn latent representations for unlabelled data, and have been successfully applied to text, image, and audio information [7]. However, there has been less work on applying auto-encoders to rating data [13]. In this paper, we propose a rating auto-encoder to learn latent features from user rating behavior, for the purpose of constructing more accurate user profiles. The generated latent user representations can then be used in recommender systems to perform personalized recommendation.

2. RELATED WORK

Recommender systems have been an active research area for more than a decade, with the main focus being recommendation approaches based on explicit ratings. Popular recommender systems applications include predicting ratings and recommending items to a user. Rating prediction is the task of predicting the rating a user will give to an item, while item recommendation is the task of recommending a set of unrated or new items to a target user [1]. Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE) are widely used to measure the accuracy of rating predictions, while precision and recall are commonly used to evaluate top-N item recommendation. For explicit ratings, both tasks are applicable, while for implicit ratings, top-N recom-

CIKM'15, October 19–23, 2015, Melbourne, Australia.
© 2015 ACM. ISBN 978-1-4503-3794-6/15/10 ...$15.00.
DOI: http://dx.doi.org/10.1145/2806416.2806633.

mendation is more applicable [1]. Recommender systems can be broadly classified into three categories: content-based, collaborative filtering ("CF"), and hybrid approaches [1]. *Neighborhood* methods [11] and *latent factor* methods [5] are the two primary collaborative filtering approaches.

Learning latent representations or latent factor models of users and items has generated recent attention, given the rapid growth of rating data in online communities [3]. Matrix factorization [5] is a state-of-the-art *latent factor* method. It jointly learns latent user and item features, and uses the dot product of the latent user and item features to predict ratings. However, there is less discussion about whether the learned latent user features themselves can be used to accurately represent user profiles. The general question of accurately learning latent user representations is still relatively unexplored. Recently, deep learning techniques [4] and neural network models [2] such as auto-encoders have been applied to various application areas to learn latent or abstract representations of words, documents, images, audio, and videos. For example, *word2vec* [10, 6] can be used to efficiently learn latent word representations based on large-scale, unlabelled text corpora. Our contribution in this work is to apply deep learning methods to user rating data, and learn latent user representations.

3. THE PROPOSED PROBABILISTIC RATING AUTO-ENCODER

To describe the proposed approach, we define some key concepts used in this paper:

- **Users:** $U = \{u_1, u_2, ..., u_{|U|}\}$ is the set of all users in an online community.
- **Items** (e.g., Products or Businesses): $P = \{p_1, p_2, ..., p_{|P|}\}$ is the set of all items rated by users in U.
- **Ratings:** $R = \{(u_i, p_j, r_{ij}) | u_i \in U, p_j \in P, r_{ij} \in \mathbb{R}\}$ is the set of all ratings that users have assigned to items, where $\mathbb{R}$ denotes the set of real numbers. Real-valued rating numbers are often binned into ordinal **Rating Categories**, denoted as $C = \{c_1, c_2, ..., c_{|C|}\}$. For example, we can represent ratings using a 5-point scale of 1 to 5, or a more coarse-grained scale such as: `negative`, `neutral`, and `positive`.

A user profile is used to describe an individual user's interests and preferences. Typically, an explicit or implicit item rating vector with numeric or binary values is used to profile the preferences or interests of a user to these items [1]. Let $\vec{u}_i$ denote the rating vector representation of user u_i, for example in the form of a set of items with ratings, $\vec{u}_i = \{(p_1, r_{i1}), (p_2, r_{i2}), ..., (p_{|P|}, r_{i|P|})\}$. For a target user u_i and target item p_j, the rating prediction task is defined as predicting the rating value r_{ij}, given a user u_i's profile.

For a given user u_i, if two items p_i and p_j are assigned the same rating score, then these two items are similar in terms of the sentiment orientation implied in the user rating information from this user's viewpoint. Based on this assumption, we propose a probabilistic auto-encoder [10] to capture the co-occurrence of items with the same rating score for the same user and learn the latent feature representation of items. For a given user and rating score, we can use the items with the same rating score from a given user as context, and use a neural probabilistic model to predict the occurrence of other items given an item in this context. For example, if user u_i rated items p_1, p_2, p_3 with the same rating score of 3, the rating auto-encoder can predict the occurrence of item p_2 or p_3 given item p_1 in this context. If the rating scores are of very fine granularity, it will result in a small context set containing very few items. To avoid data sparsity, we bin the rating scores and map them to rating categories with coarser granularity.

Let $P_{ix} \subseteq P$ denote the items in the context of user u_i and rating category c_x. The objective of the rating auto-encoder is then to maximize the average log probability:

$$\frac{1}{|P_{ix}|} \sum_{p_s \in P_{ix}} \sum_{p_t \in P_{ix} \setminus \{p_s\}} \log \Pr(p_t | p_s) \qquad (1)$$

Inspired by the *skip-gram* model of *word2vec* [10], the rating auto-encoder contains two layers: an input and output layer. The input to the rating auto-encoder is the vector representation of an item $p_s \in P_{ix}$ with random initialization, while the output of the rating auto-encoder is every other item in the same context $p_t \in P_{ix} \setminus \{p_s\}$. Let W be the weight matrix between the input layer and output layer; W is randomly initialized and shared across all contexts. Let y denote the un-normalized log-probability for each output item p_l, which can be computed as:

$$y = W \cdot p_s \qquad (2)$$

We use a multi-class classifier (e.g., softmax) to conduct the prediction task:

$$\Pr(p_t | p_s) = \frac{e^{y_{p_t}}}{\sum_{l=1}^{|P|} e^{y_{p_l}}} \qquad (3)$$

We use the squared difference error as the loss function, stochastic gradient descent to train the auto-encoder model, and back propagation to obtain the gradient to update each input vector of item p_s and weight matrix W. After convergence over the training data, items with similar ratings are mapped to a similar position in the vector space. We generate latent representations of both items and users, as detailed below.

The **item representation** represents each item's latent rating behavior by users. Let $L = \{1, 2, ..., d\}$ denote the latent factor, v_{ja} denote the relevance of item p_j to latent factor $l_a \in L$ (i.e., the value of dimension l_a of the learned or updated vector of item p_j), and $2^{L \times \mathbb{R}}$ denote the power set of $L \times \mathbb{R}$. The relationship between an item and a set of latent factors can then be defined as the mapping $\mathcal{R}^p : P \to 2^{L \times \mathbb{R}}$, such that $\mathcal{R}^p(p_j) = \{(p_j, v_{ja}) | l_a \in L\}$. $\mathcal{R}^p(p_j)$ is called the item representation of item p_j.

Based on the item representation and the rating profile of each user, we can derive the latent representation profile of a given user u_i in a rating category $c_x \in C$. The **user representation** represents the latent rating behavior of each user. Let $w_{ia}^{c_x}$ denote the value of how much user u_i is interested in latent factor $l_a \in L$ in rating category $c_x \in C$. The relationship between a user and a set of latent factors in rating category $c_x \in C$ can then be defined as the mapping $\mathcal{R}^{u,c_x} : U \to 2^{L \times \mathbb{R}}$, such that $\mathcal{R}^{u,c_x}(u_i) = \{(l_a, w_{ia}^{c_x}) | l_a \in L\}$. $\mathcal{R}^{u,c_x}(u_i)$ is the user representation of u_i in rating category $c_x \in C$, and $\mathcal{R}^u = \{\mathcal{R}^{u,c_1}(u_i), \mathcal{R}^{u,c_2}(u_i), ..., \mathcal{R}^{u,c_{|C|}}(u_i)\}$ is the user representation of u_i.

We adopt **mean pooling** [7] to get the summary statistics of the latent features of u_i in category $c_x \in C$. The weight

$w_{ia}^{c_x}$ can be calculated as:

$$w_{ia}^{c_x} = \frac{\sum_{p_j \in P_{ix}} v_{ja}}{|P_{ix}|} \tag{4}$$

Based on user profiles, we use the *neighborhood* based collaborative filtering approach to find a set of similar users, then perform rating prediction based on neighboring users' ratings. This will be discussed in the following section.

4. PERSONALIZED RECOMMENDER SYSTEMS

The *neighborhood* based collaborative filtering approach is a common method for making recommendations. We apply the generated user profiles in *neighborhood* based recommender systems to make rating predictions. Neighborhood formation is the task of generating like-minded peers (i.e., the k nearest neighbors, "k-NN") for a target user $u_i \in U$. The distance or similarity measure can be calculated through various kinds of proximity computing approaches such as cosine similarity or Pearson's correlation. The more accurate a user profile is, the higher quality the user neighborhood.

Typically, the similarity of two users u_i and u_j can be measured by the similarity of their user profiles (i.e., user representations). Cosine is used to measure the similarity of two user latent representation vectors. The similarity of u_i and $u_j \in U$ can be calculated based on the maximum similarity of their representations over all the rating categories:

$$sim(u_i, u_j) = \max_{c_x \in C}\{cosine(\mathcal{R}^{u,c_x}(u_i), \mathcal{R}^{u,c_x}(u_j))\} \tag{5}$$

The neighborhood of user u_i is denoted as $\mathcal{N}(u_i) = \{u_j | u_j \in maxK_{u_j \in U}\{sim(u_i, u_j)\}\}, u_j \in U$, where $maxK\{\}$ returns the top-k most similar users to u_i. For each target user u_i, the prediction score of how much u_i will be interested in item p_j is calculated by considering the similarities between user u_i and those users who are neighbors of u_i and have rated item p_j [11]:

$$\hat{r}_{ij} = b_{ij} + \frac{\sum_{u_z \in \mathcal{N}_{u_i}} sim(u_i, u_z)(r_{zj} - b_{zj})}{\sum_{u_z \in \mathcal{N}_{u_i}} sim(u_i, u_z)} \tag{6}$$

where $b_{ij} = \mu + b_{u_i} + b_{p_j}$, μ is the average rating score of all known ratings, b_{u_i} is the observed rating deviation of user u_i, $b_{u_i} = \bar{r}_{u_i} - \mu$, and $\bar{r}_{u_i}$ denotes the average rating of user u_i. b_{p_j} is the observed rating deviation of item p_j, $b_{p_j} = \bar{r}_{p_j} - \mu$, where $\bar{r}_{p_j}$ denotes the average rating of item p_j. $b_{zj} = \mu + b_{u_z} + b_{p_j}$, where b_{u_z} is the observed rating deviation of user u_z.

5. EXPERIMENTS

To evaluate the effectiveness of the proposed approach, we conducted experiments on a real-world rating dataset from `yelp.com`.

5.1 Dataset

The Yelp dataset is a customer review dataset.[1] Each user rates businesses with a score ranging from 1 to 5. To reduce sparseness in the dataset, we filtered out those users who have rated no more than 3 items, and those items that have been rated by no more than two users. The dataset includes 37,915 users, 40,021 items (i.e., businesses), and 713,291 ratings. For each user, we randomly selected 80% of ratings as training data, and 20% of ratings as test data. The task is to predict ratings for the test set based on the training data. The descriptive statistics of this dataset are shown in Table 1.

[1]`http://www.yelp.com.au/dataset_challenge`

Table 1: The composition of the Yelp dataset

	Training Set	Test Set
Users	37,915	37,915
Items	38,977	20,413
Ratings	599,546	113,745

5.2 Experimental Setup

We use root mean squared error ("RMSE") to measure the accuracy of rating prediction:

$$\text{RMSE} = \sqrt{\sum_{r_{ij} \in \text{TestSet}} (r_{ij} - \hat{r}_{ij})^2 / |\text{TestSet}|} \tag{7}$$

We compared the RMSE value of the proposed approaches (i.e, *RA-implicit* and *RA-explicit*) with *neighborhood* methods over the Yelp dataset. To evaluate the effectiveness of user profiles, we compare the following user profile representations using a standard user-based k-NN collaborative filtering approach:

- ***RA-implicit***: latent user profiles generated by the proposed implicit rating auto-encoder. For this approach, the size of the rating space is 1 (i.e., $|C| = 1$). That is, all user ratings in the training set are mapped to one rating category, meaning that $C = \{1\}$. If a rating is greater than 0, then the value of the rating is 1.
- ***RA-explicit***: latent user profiles generated by the proposed explicit rating auto-encoder. For this approach, the size of the rating space is 3 (i.e., $|C| = 3$). That is, all user ratings in the training set are mapped to three rating categories, namely $C = \{1, 2, 3\}$. If the original rating is greater than 0 and smaller than 3, then a category of 1 is assigned; if a rating equals 3, then a category of 2 is assigned; and if a rating is greater than 3, then a category of 3 is assigned.
- ***Random***: latent user profiles generated by assigning random values for each latent factor (based on the same dimensionality d as *RA-implicit* and *RA-explicit*). The purpose of this method is to determine whether any gains to the proposed approaches (*RA-implicit* and *RA-explicit*) are due simply to the expressivity of the representation (in which case *Random* should perform comparably to the other two methods), or the representation learning (in which case *Random* should perform worse than *RA-implicit* and *RA-explicit*).
- ***MF***: latent user profiles generated by the matrix factorization approach [5].
- ***CF***: explicit ratings, in the form of the original ratings.

For all of *RA-implicit*, *RA-explicit*, *Random* and *MF*, we set the dimensionality parameter d to 100.

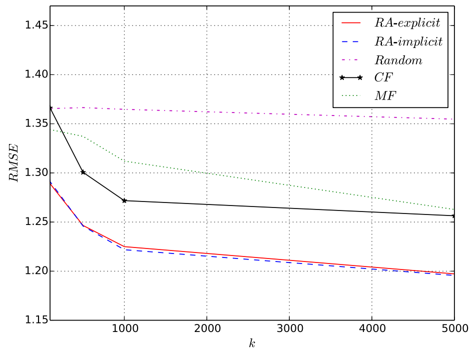

Figure 1: The RMSE of rating predictions for the different approaches over the Yelp dataset

5.3 Results

Figure 1 shows the RMSE results of the different approaches under varying values of k. We can see that *RA-implicit* performs slightly better than *RA-explicit*, while *RA-explicit* and *RA-implicit* achieved better performance (i.e., lower RMSE scores) than the other models. Given that the accuracy of rating prediction relies on the quality of the target user representation, this indicates that *RA-implicit* and *RA-explicit* result in better user representations. Despite having the same dimensionality as *RA-implicit* and *RA-explicit*, *Random* performed the worst, indicating that our results are not simply an artefact of the more expressive representation, but rather due to the quality of the learned representation. The user profiles generated by the proposed rating auto-encoder are also more accurate than the latent factor user profiles generated by *MF*.

Based on a one-tailed paired t-test, both *RA-explicit* and *RA-implicit* are significantly better than the other *neighborhood* baselines ($p < 0.005$), but there is no significant difference between the two *RA* methods ($p > 0.1$). Because the dataset is sparse (81.2% users have provided less than 5 ratings), coarser granularity leads to higher co-occurrence of items in the same context, explaining why *RA-implicit* is slightly better than *RA-explicit*.

6. CONCLUSION

In this paper, we have proposed a probabilistic rating auto-encoder to conduct unsupervised feature learning for learning latent user profiles from large-scale rating data. The rating auto-encoder learns item similarity based on analysis of items which have been given similar ratings by a given user. Based on the generated latent item representations, we generate latent user representations in each rating category. After that, we applied the generated latent user profiles to *neighborhood* based collaborative filtering recommender systems to predict rating scores. The experiments were conducted on real-life rating data from `yelp.com` shows that the latent user profiles generated by the proposed rating auto-encoder are able to find more similar neighboring users without further fine tuning or optimization, and make more accurate rating predictions than traditional rating-based user profiles and latent user profiles generated by state-of-the-art latent factor learning models such as matrix factorization.

In future work, we plan to explore how to learn latent user features from multi-modality social data such as both rating and user-created text information. We are also interested in combining the proposed approach with matrix factorisation for rating prediction, to see whether further gains can be achieved.

Acknowledgements

This research was supported by the Australian Research Council. The authors would like to acknowledge the support of Pitchwise (`www.pitchwi.se`), and valuable feedback from the anonymous reviewers and Joel Nothman.

7. REFERENCES

[1] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *TKDE*, 17(6):734–749, 2005.

[2] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3:1137–1155, Mar. 2003.

[3] X. Chen and X. Lin. Big data deep learning: Challenges and perspectives. *IEEE Access*, 2:514–525, 2014.

[4] G. Hinton and R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, July 2006.

[5] Y. Koren, R. Bell, and C. Volinsky. Matrix Factorization Techniques for Recommender Systems. *Computer*, 42(8):30–37, Aug. 2009.

[6] Q. V. Le and T. Mikolov. Distributed representations of sentences and documents. *CoRR*, abs/1405.4053, 2014.

[7] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, May 2015. Insight.

[8] H. Liang, Y. Xu, Y. Li, R. Nayak, and X. Tao. Connecting users and items with weighted tags for personalized item recommendations. In *Proceedings of the 21st ACM Conference on Hypertext and Hypermedia (Hypertext 2010)*, pages 51–60, 2010.

[9] H. Liang, Y. Xu, D. Tjondronegoro, and P. Christen. Time-aware topic recommendation based on micro-blogs. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management (CIKM 2012)*, pages 1657–1661, Maui, USA, 2012.

[10] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.

[11] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the Tenth International World Wide Web Conference (WWW10)*, pages 285–295. ACM, 2001.

[12] J. Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, 2015. Published online 2014; based on TR arXiv:1404.7828 [cs.NE].

[13] H. Wang, N. Wang, and D. Yeung. Collaborative deep learning for recommender systems. *CoRR*, abs/1409.2944, 2014.