Collaborative Deep Learning for Recommender Systems

Saurabh Kapur 2015087[1] and Sumeet Bharadwaj 2015182[2]

[1] IIIT Delhi
saurabh15087@iiitd.ac.in
[2] IIIT Delhi
sumeet15182@iiitd.ac.in

1 Introduction

An increasing number of companies like Amazon, Netflix and others are utilizing recommendation systems (RS) to better serve their users. Better user recommendation helps these companies retain users. Due to the ever increasing use of e-commerce and online streaming services, building better recommendation systems is imperative. Existing methods for RS can be roughly categorized into three classes: content-based methods, collaborative filtering (CF) based methods, and hybrid methods. Content-based methods make use of user profiles or product descriptions for recommendation. CF-based methods use the past activities or preferences, such as user ratings on items, without using user or product content information. Hybrid methods seek to get the best of both worlds by combining content-based and CF-based methods.

Hybrid method can also be divided into 2 sub categories: Loosely coupled and tightly coupled methods. In loosely coupled method the rating information does not provide any feedback and the auxiliary information is processed only once which is used as features for collaborative filtering method. On the contrary in tightly coupled methods there is a two way interaction. The rating information also provides a feedback which helps in learning the features better. Using the tightly coupled method we can automatically learn better features from the auxiliary information and with better features the result of CF method also improves.

We implement a hierarchical Bayesian model called Collaborative Deep Learning [1](CDL) as a novel tightly coupled method for recommendation system. This method allows two interactions between the rating information and the auxiliary information.

2 Dataset

Collaborative Deep learning is a hybrid method which uses both content information and rating information. Hence, the dataset we use contains information of the item in the form of some text description and the rating provided by the users. Using the user ratings we build a rating matrix. This matrix contains binary values - 1 if the user likes the item and 0 if the user dislikes the item. We used 4 datasets in our project: citeulike-a[5], citeulike-t[5], Last.fm[4] and FlickScore dataset[2].

Citeulike [5]: CiteULike is a web service which allows users to save and share citations to academic papers. Based on the principle of social bookmarking, the site works to promote and develop the sharing of scientific references among researchers. This is similar to how other services like Flickr, Furl and Delicious catalogue photographs and web pages respectively. Scientists are able to share citation information using CiteULike. The CiteULike dataset is a collection of articles including abstracts, titles, and rating information for each article.

FlickScore [2]: FlickScore is a movies dataset. It consists of the movie and related user supplied information like rating. For movies it consists of data like Description, languages, Release date, genre and such. However, for this project we only consider movie description and user-item ratings. In the FlickScore dataset we only use the movies for which the description is available.

Last.fm [4]: Last.fm provides dataset for music recommendations. For each user in the dataset it contains a list of their top most listened to artists including the number of times those artists were played. It also includes user applied tags which could be used to build a content vector. This is a big dataset but for our project we take a subset of this dataset. We use binary values in the rating matrix which stores the users against the artists that they have listened to.

Table 1. Number of users and items in the dataset

Dataset	Users	Items
citeulike-a	5551	16980
citeulike-t	7947	25975
FlickScore	924	1752
Last.fm	1226	285

3 CDL method

As discussed before, there are two phases in learning the user-item relation. In the proposed CDL method[1] first phase uses a Stacked Denoising Autoencoder(SDAE) for learning descriptions. Probabilistic matrix factorization(PMF) is used for predicting rating using the output generated by SDAE as part of the second phase. SDAE, like a feedforward neural network, is used for learning representations (encoding) of the input data. The model tries to learn the encoding by predicting the input itself. Discussed in the implementations section.

Figure 1 shows the CDL model used in this paper. The part inside the dashed rectangle represents an SDAE. Input to this model is a bag of words vector that

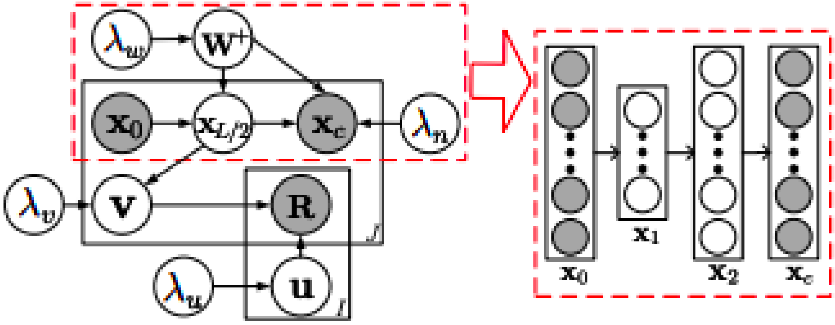

Fig. 1. CDL model

is pre-computed. The SDAE takes $\mathrm{X_0}$ (BoW vector) and encodes it into another vector $\mathrm{X_{L/2}}$. This encoded vector is then passed into the Probabilistic Matrix Factorization model (PMF). Since this method only uses the information of the items therefore we do not consider the user information. The encoded output of the SDAE becomes the input to the item branch of the PMF. We initialize the user latent vector as follows

$$u_i \sim \mathcal{N}(0, \lambda_u^{-1} I_k) \tag{1}$$

Where λ_u is a hyper-parameter. The rating information provides feedback to SDAE which helps in the learning process. After minimizing the loss we get latent item vectors $\mathrm{v_j}$ and user vectors $\mathrm{u_i}$. Then the rating is predicted as follows,

$$R_{ij} \approx u_i^T v_j \tag{2}$$

4 Our Implementation

We use the CDL model implementation[3] in MxNet (Amazon's Deep Learning library). As mentioned in the previous sections, this model uses the bag of words representation of the any text description. It is built using the frequency of the words in the vocabulary for a given text. In [1] the vocabulary is formed by using tfidf values of the words. The citeulike dataset provides the vector representation of the abstracts.

For the FlickScore dataset, to find the bag of words representation we first build the vocabulary. As part of the data preprocessing we first remove the stopwords and punctuations in the movie descriptions. After preprocessing, we found out the tfidf values for the words to find the most relevant words. We sorted the words in decreasing order of their tfidf values. After ranking the words we take the top 4000 most relevant words to form the vocabulary.

For Last.fm dataset[4] we take a subset of the dataset. We use binary values in the rating matrix which contains what users have listened to what artists. The dataset does not contain the text description of the artist so we collect the artist information by scraping the web. We found the artist bio in Last.fm website[6]. To get the bio we implemented a web scrapping script using python and beautiful soup library. After getting the bio for each artist we find the vocabulary of the documents by finding the tfidf values of each word and ranking the words based on the tfidf values. From the ranked words we use the top 4000 words as our vocabulary for the dataset. As part of data preprocessing we remove stop words and punctuations from the text. Table 2 shows the vocabulary sizes of each dataset.

Table 2. Vocabulary size

Dataset	Vocabulary size
citeulike-a	8000
citeulike-t	20000
FlickScore	4000
Last.fm	4000

After finding the vocabulary we create the feature vector for each item by finding the count of each words in the description. This feature vector becomes the input to our CDL model.

5 Evaluation metric

For the citulike dataset we compute the mean average precision(mAP) to compare the results with paper[1] as one of their evaluation metric is mAP. For the FlickScore dataset we use the MAE and RMSE values to compare the result. For Last.fm dataset we use mAP and Recall@100.

6 Results

6.1 Citeulike

We use the same hyper parameters mentioned in the paper. We found the results of sparse and dense settings for the citeulike-a dataset. In the sparse setting only one article in the user's library is in the training set rest all are in the testing dataset. Table 3 compares our sparse setting results with the paper[1]. To find the mAP we take the top 500 recommendations.

In the dense setting we get mAP value of 0.104 for the citeulike-a dataset. In the paper they have calculated the mAP for dense setting. Table 4 shows the top 5 recommendation of user 4 given the training data.

Table 3. Results for sparse setting in terms of mAP

’	citeulike-a(mAP)	citeulike-t(mAP)
Our implementation	0.048	0.054
Paper’s result	0.0514	0.0453

Table 4. top 5 Recommended articles

Recommendations (Correct Ones Marked in bold)
The Semantic Web
A Translation Approach to Portable Ontology Specifications
The Semantic Web Revisited
Towards principles for the design of ontologies used for knowledge sharing
Semantic integration: a survey of ontology-based approaches

6.2 FlickScore

We calculate the MAE and RMSE to compare the result of CDL. The mAP value for flickScore is 0.018. Table 5 shows the MAE and RMSE values.

Table 5. MAE and RMSE values

Method	MAE	RMSE
CDL	0.319	0.405

6.3 Last.fm

We calculate the mAP value for this dataset and recommend artist to users. Table 6 shows the mAP and recall@100 values. Table 7 shows the recommendations given the training data.

Table 6. mAP and recall@200 value

Method	mAP	recall@100
CDL	0.129	0.647

Table 7. top 5 Recommended artists

Artists in the Training Sets
blind guardian maria mena red hot chili peppers subway to sally
Recommendations (Correct Ones Marked in bold)
metallica system of a down **rammstein** red hot chili peppers nightwish

References

1. Wang H., Wang N., Yeung D.Y.: Collaborative deep learning for recommender systems. KDD. (2015)
2. Majumdar A., Agarwal P., Verma R.: Indian regional movie dataset for recommender systems. Conf. (2017)
3. Code: https://github.com/js05212/MXNet-for-CDL
4. Last.fm dataset - https://gist.github.com/victorkohler/0931d181ef126e0740d8aac6933f13f4, https://medium.com/radon-dev/item-item-collaborative-filtering-with-binary-or-unary-data-e8f0b465b2c3
5. CiteULike dataset - http://www.citeulike.org/faq/data.adp
6. Last.fm artist bio dataset - https://www.last.fm/music