Restricted Boltzmann Machines for Collaborative Filtering

Article Review

MEKHZOUM Hamza
University of Rouen
hamza.mekhzoum@univ-rouen.fr

BOUSSOUF Omar
University of Rouen
omar.boussouf@univ-rouen.fr

HAMAMA Krimo
University of Rouen
krimo.hamama@etu.univ-rouen.fr

ABSTRACT

A well-known technique used mainly for designing recommender systems is collaborative filtering. In this paper, we'll discuss the Restricted Boltzmann Machine (RBM) an algorithm used for such a problematic. The paper show that RBM can be successfully applied to the Netflix data set, containing over 100 million user/movie ratings with efficient learning and great results. The paper also show that the system achieved an error rate that is well over 6% better than the score of Netflix's own system. The model is trained and evaluated based on the number of hidden units, learning rates, and activation functions. Inspired by the paper, we'll use this model to train our dataset to recommend users a set of specified movies.

General Terms

Restricted Boltzmann machine, collaborative filtering; recommender systems; similarity score; popularity score.

Keywords

Restricted Boltzmann machine; Artificial Intelligence; collaborative filtering; Machine learning, recommender systems.

INTRODUCTION

Recommender systems (RSs) significantly enhance the users' experience when accessing online services. Amazon, Facebook, and Netflix extensively use RSs to attract users by advising their products and services. RSs are used to generate lists of suggestions using approaches such as collaborative filtering (CF), content-based filtering, or hybrid methods [1].

The CF methods utilize past activities and preferences of like-minded users to predict a user's interest and generate recommendations. They use the nearest neighbour approach based on activities and ratings of previous users. Content-based methods rely on users' profiles and descriptions of products for advertising [2] while hybrid methods combine both techniques [3].

Early CF approaches used singular value decomposition (SVD), which relies on matrix factorization. Few attempts have been made to develop RS learning models using collaborative filtering.

Based on the article [4] RBM slightly outperform carefully tuned SVD models and, overall outperforming Netflix recommender system.

LITERATURE REVIEW

In this chapter, various works carried out in the literature on Collaborative filtering for recommender systems are reviewed.

In 2007 a paper that outperformed all previous methods in collaborative filtering was published in aim for a competition launched by Netflix to elaborate a new algorithm for their recommender system, Netflix made their data free to all researchers, data contained over 100 million user/movie ratings from 1998 to 2005. In the paper R.Salakhutdinov et al [4] showed incredible results outperforming both SVD approach and Netflix own system.

A wide and deep learning model [5] that combines the strength of feature memorization of the linear model and feature generalization of the deep learning model. It is proposed for recommending applications for the Google Play store.

The deep neural network used for advertising YouTube videos performs a candidate generation followed by ranking [6]. Authors used a dataset consisting of 106 videos and 106 search requests (tokens). They considered both positive (clicked) and negative (unclicked) selections of movies shown to a user on a single web page. Selections are weighted by the duration of watch time.

A collaborative deep learning model framework [7] jointly performs deep representation learning for the content information and collaborative filtering for the ratings. The authors define a performance metric to recommend the top M items. The relational collaborative topic regression develops a hierarchical Bayesian model that integrates the user-item feedback.

GENERAL WORKFLOW OF THE SYSTEM

The RBM is a generative stochastic artificial neural network that learns a probability distribution over a set of inputs. It employs gradient descent approximation algorithms such as contrastive divergence (CD). The RBM has been used for feature extractions in supervised learning algorithms. It belongs to energy-based models and consists of Bernoulli-valued (binary) hidden and visible units, as shown in Fig. 1. The matrix of weights W represents the strength of connections between visible and hidden units Vi, and Hj, respectively. It includes bias weights (offsets) for these units. Increasing the number of hidden units may improve performance of the RBM model.

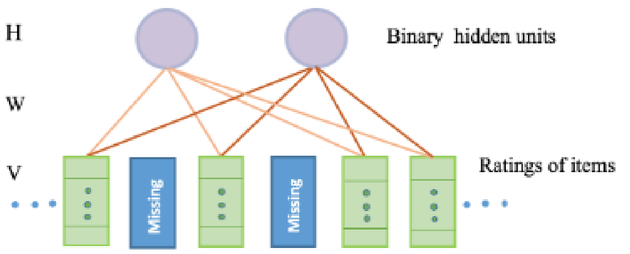

Fig. 1. The RBM model: Each visible unit corresponds to an item that is rated. Items that are not rated are considered missing.

In the dataset we may have empty values, in such a condition we can use the conditional RBM. Many users view items without providing feedback and, hence, their ratings are unknown. Furthermore, the test dataset may contain several ratings from a user while the training dataset may have fewer ratings for the same item. If a user views and rates an item, this may lead to his/her choice of similar items. This additional source of information about users and items is considered in the conditional restricted Boltzmann machine (CRBM) model. The CRBM is a probabilistic model that considers in visible layer both rated and unrated items along with additional information. It has been successfully used for modelling temporal data such as motion capture in video sequences, collaborative filtering, and classification. Using a conditional probability of rated and unrated movies improves the general learning of the system and reduce error.

Since Boltzmann Machines are energy-based models and a joint configuration, (v,h) of the visible and hidden units has an energy given by:

$$E(\mathbf{v},\mathbf{h}) = -\sum_{i\in visible} a_i v_i - \sum_{j\in hidden} b_j h_j - \sum_{i,j} v_i h_j w_{ij}$$

where vi, hj, are the binary states of the visible unit i and hidden unit j, ai, bj are their biases and wij is the weight between them.

In the learning process, the parameter updates required to perform gradient ascent in the log-likelihood can be obtained from:

$$\Delta W_{ij}^k = \epsilon \frac{\partial \log p(\mathbf{V})}{\partial W_{ij}^k} = \\ = \epsilon\left(<v_i^k h_j>_{data} - <v_i^k h_j>_{model} \right)$$

where "e" is the learning rate. The expectation <vki hj>data defines the frequency with which movie i with rating k and feature j are on together when the features are being driven by the observed user-rating data from the training set. The expectation <·> model cannot be computed analytically in less than exponential time. MCMC methods (Neal, 1993) [8] can be employed to approximate this expectation. These methods, however, are quite slow and suffer from high variance in their estimates.

To avoid computing <·> model, they followed an approximation to the gradient of a different objective function called "Contrastive Divergence" (CD) (Hinton, 2002) [9]:

$$\Delta W_{ij}^k = \epsilon(<v_i^k h_j>_{data} - <v_i^k h_j>_T)$$

The expectation <·> T represents a distribution of samples from running the Gibbs sampler. It was shown

(Hinton, 2002) that CD learning is quite efficient and greatly reduces the variance of the estimates used for learning.

RESULTS

In the experimental phase, we implemented the RBM algorithm in a dataset taken from grouplens a research lab in the department of Computer science at the University of Minnesota specializing in recommender systems. The dataset contains 1M data of 3952 Films noted by 6040 Users.

In our algorithm, we first initialized our weights then, we performed two main steps, Gibbs sampling (MCMC algorithm) to generate posteriori samples and contrastive divergence to calculate our gradient.

In the learning phase, we used the following Hyperparameters:

- Number of NN in the visible layer: 3952
- Number of NN in the hidden layer: 100
- Batch size: 10
- Number of Epoch: 50
- Learning rate: 0.01

To evaluate our RBM we choosed Average distance instead of RMSE as it's more intuitive, with this metric we obtained the following results in the training phase:

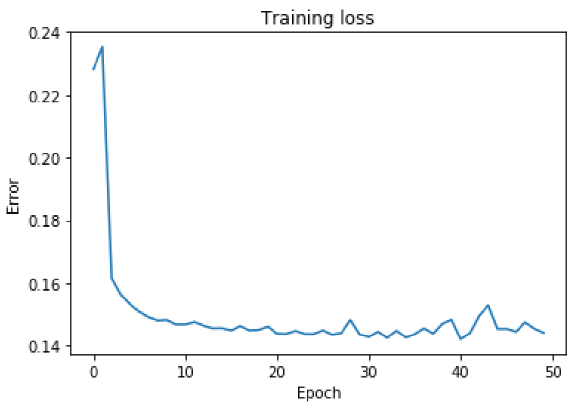

Fig. 2. The training loss in 50 epochs.

And in the test phase we obtained: tensor (0.2149) loss

Upon launching our algorithm to recommend to a specific user a set of movies, we had the following results, we displayed only the first 10 recommended movies. Full results can be found in the Jupyter file:

<table>
<tr><th>MovieID</th><th>Title</th><th>Genres</th><th>List Index_x</th><th>Recommandation</th><th>Rating</th></tr>
<tr><td>1269</td><td>Arsenic and Old Lace (1944)</td><td>Comedy|Mystery|Thriller</td><td>1249</td><td>1.000000</td><td>NaN</td></tr>
<tr><td>1209</td><td>Once Upon a Time in the West (1969)</td><td>Western</td><td>1191</td><td>1.000000</td><td>NaN</td></tr>
<tr><td>3417</td><td>Crimson Pirate, The (1952)</td><td>Adventure|Comedy|Sci-Fi</td><td>3348</td><td>1.000000</td><td>NaN</td></tr>
<tr><td>1197</td><td>Princess Bride, The (1987)</td><td>Action|Adventure|Comedy|Romance</td><td>1179</td><td>1.000000</td><td>5.0</td></tr>
<tr><td>3470</td><td>Dersu Uzala (1974)</td><td>Adventure|Drama</td><td>3401</td><td>1.000000</td><td>NaN</td></tr>
<tr><td>1238</td><td>Local Hero (1983)</td><td>Comedy</td><td>1219</td><td>1.000000</td><td>NaN</td></tr>
<tr><td>1609</td><td>187 (1997)</td><td>Drama</td><td>1567</td><td>1.000000</td><td>NaN</td></tr>
<tr><td>1264</td><td>Diva (1981)</td><td>Action|Drama|Mystery|Romance|Thriller</td><td>1244</td><td>1.000000</td><td>NaN</td></tr>
<tr><td>1306</td><td>Until the End of the World (Bis ans Ende der W...</td><td>Drama|Sci-Fi</td><td>1286</td><td>1.000000</td><td>NaN</td></tr>
<tr><td>2639</td><td>Mommie Dearest (1981)</td><td>Drama</td><td>2570</td><td>0.999999</td><td>NaN</td></tr>
</table>

Fig. 3. Recommendation for a random user a set of specific movies.

As shown in the results, the algorithm ran smoothly and performed good recommendations for the random user, specifying a set of movies never seen by the random user, according to the collaborative filtering that we performed. For example, the user already saw the 4th movie. Thus, the system recommended that movie with a percentage of 100%, also the random user tends to watch drama and comedy movies a lot, and as seen in the results, the algorithm recommended similar genres of films, which means that the algorithm was well performed.

CONCLUSION

The following paper gives a glimpse of information on RBM. The recommender systems based on predictions currently play an important role in advertisements of Internet services and products. a class of two-layer undirected graphical models (RBM's), suitable for modelling tabular or count data, and presented efficient learning and inference procedures for this class of models. It was also demonstrated that RBM's can be successfully applied to a large dataset containing over 100 million user/movie ratings. As for today, there're many other approaches much better than RBM based on reinforcement learning and produces incredible results.

REFERENCES

[1] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutierrez, "Recommender systems survey," J. Knowledge Based Systems, vol. 46, pp. 109–132, July 2013.

[2] K. Lang, "Newsreader: learning to filter Netnews." in Proc. Int. Conf. Mach. Learning, Tahoe City, CA, USA, July 1995, pp. 331–339.

[3] X. Wang and Y. Wang, “Improving content-based and hybrid music recommendation using deep learning,” in Proc. ACM Int. Conf. Multimedia, Orlando, FL, USA, Nov. 2014, pp. 627–636.

[4] R. Salakhutdinov, A. Mnih, and G. Hinton, “Restricted Boltzmann machines for collaborative filtering,” in Proc. Int. Conf. Mach. Learning, Corvallis, OR, USA, June 2007, pp. 791–798.

[5] H. T. Cheng, L. Koc, J. Harmsen, et al., “Wide and deep learning for recommender systems,” in Proc. ACM Workshop on Deep Learning for Recommender Syst., Boston, MA, USA, Sept. 2016, pp. 7–10.

[6] P. Covington, J. Adams, and E. Sargin, “Deep neural networks for YouTube recommendations,” in Proc. ACM Conf. Recommender Syst., Boston, MA, USA, Sept. 2016, pp. 191–198.

[7] H. Wang, N. Wang, and D. Y. Yeung, “Collaborative deep learning for recommender systems,” in Proc. ACM Int. Conf. Knowledge Discovery and Data Mining, Sydney, Australia, Aug. 2015, pp. 1235–1244.

[8] Neal, R. M. (1993). Probabilistic inference using Markov chain Monte Carlo methods (Technical Report CRG-TR-93-1). Department of Computer Science, University of Toronto.

[9] Carreira-Perpinan, M., & Hinton, G. (2005). On contrastive divergence learning. 10th Int. Workshop on Artificial Intelligence and Statistics (AISTATS’2005).