Movie Recommendation System based on Metadata and User Ratings

Jiaxi Chen, Yongshang Wu, and Ziran Zhang

Problem Definition

- **Problem**: Give movie recommendation to users based on their previous movie ratings and movie contents.
- **Goal**: Build an end-to-end recommendation system that predicts users' potential ratings on movies, which could perform reasonably well even starting with pretty sparse data, by incorporating intrinsic metadata of movies.

Existing Works

- **Collaborative Filtering** [4] [5] is a popular and powerful technique employed by many recommendation systems, such as Google News Personalization [1] and Amazon.com [7]. Matrix factorization [3] methods like Principal Component Analysis and Singular Value Decomposition has been widely employed to tackle this problem.

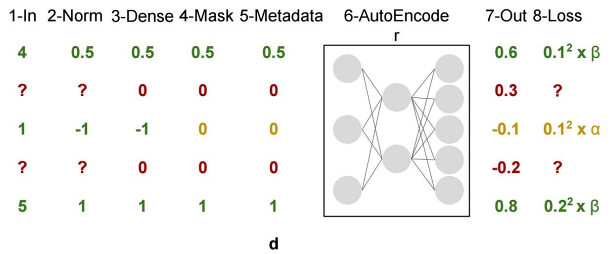
  Recently, as the emerging of deep learning techniques, researchers also did a few attempts on recommendation system based on both Neural Network models [6] and Recurrent Network models [8].
- **Drawbacks**: these techniques suffer from sparsity problem, also known as the 'cold start' issue. It also takes lots of efforts or even is impossible to incorporate any domain knowledge or side information.

Dataset

- **Source**: The dataset we use is 'The Movie Dataset' [2] from kaggle.com, which consists of more than 26,000,000 ratings from over 270,000 users on 40,000 movies. This dataset also comes with textual movie metadata, e.g. overviews, keywords and genres.
- **Preprocessing**: The density of raw data is 0.25%. We first select out users/movies that give/receive more than 300 ratings, then keep movies with all metadata, leaving us 2,095 movies and 7,941 users with ~1 million ratings, i.e. 7.2% density.
- **Input**: The model inputs are a (2095, 7941)-shaped matrix R constructed out of the preprocessed ratings, and a (2095, 300)-shaped metadata matrix obtained by averaging word vectors of textual metadata.

AutoEncoder

- Our approach builds on an autoencoder to predict full movie rating vectors $r_{i,:}$ (i-th row from matrix R) from incomplete vectors.
- Ratings are normalized to [-1, 1] scale and the output layer uses a tanh activation.
- We will treat unknown values as 0's and exclude them when computing loss.
- Metadata is incorporated by concating the original movie vector with it's side info.



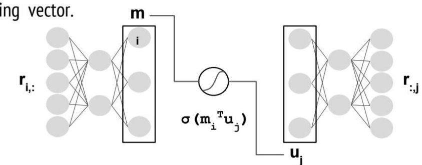
- The denoising autoencoder (DAE) loss function:

$$\mathcal{L}_{\mathrm{DAE}}(R_{\cdot j}, \hat{R}_{\cdot j}) = \alpha \sum_{i \in \mathcal{E}(R_{\cdot j}) \cap \mathcal{M}(R_{\cdot j})} \left(\hat{R}_{ij} - R_{ij}\right)^2 + \beta \sum_{i \in \mathcal{E}(R_{\cdot j}) \setminus \mathcal{M}(R_{\cdot j})} \left(\hat{R}_{ij} - R_{ij}\right)^2 + \lambda \|\mathbf{W}\|_2^2$$

- By masking known values and employing DAE loss, we emphasize the autoencoder's ability to **predict** known values instead of focusing on **recovering**.

Score Model

- Inspired by word2vec [9], we tried training a dense embedding of movies and users. A confidence score is computed as dot product of movie embedding vector and user embedding vector.



- The Score Model is essentially a binary classification model. To support that, we transform the ratings to 1 (like) for rating >= 4 and -1 (unlike) for the rest of them. As a result, logistic loss is used:

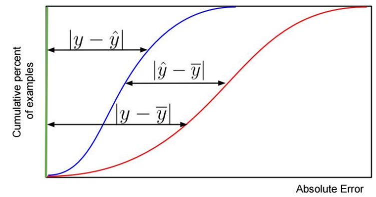
$$\mathcal{L}_{\mathrm{Score}}(R_{\cdot j}, \hat{R}_{\cdot j}) = \frac{1}{2} \sum_{i,j \in \mathcal{E}(R)} -y_{i,j} \log \hat{y}_{i,j} - (1 - y_{i,j}) \log (1 - \hat{y}_{i,j})$$

Result

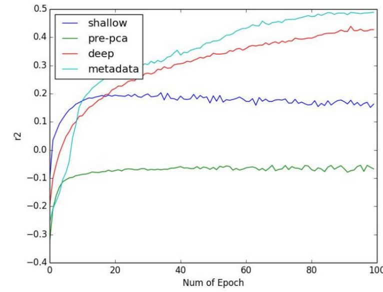
- Evaluation Metric

  We use coefficient of determination (r^2) as the metric of performance evaluation over our models:

$$R^2 = 1 - \frac{\sum |y - \hat{y}|^2}{\sum |y - \overline{y}|^2}$$



- AutoEncoder Architecutre Comparison



- Accuracy Comparison between AutoEncoder and Score Model

Model	AutoEncoder	Score Model
Accuracy	74.31%	71.01%

Reference

[1] Abhinandan S Das, Mayur Datar, Ashutosh Garg, and Shyam Rajaram. Google news personalization: scalable online collaborative filtering. In Proceedings of the 16th international conference on World Wide Web, pages 271– 280. ACM, 2007. [2] Martin E Fabien D, Rounak B. The movie dataset. https: //www.kaggle.com/rounakbanik/the-movies-dataset, 2017. [3] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009. [4] Will Hill, Larry Stead, Mark Rosenstein, and George Furnas. Recommending and evaluating choices in a virtual community of use. In Proceedings of the SIGCHI conference on Human factors in computing systems, pages 194–201. ACM Press/Addison-Wesley Publishing Co., 1995. [5] Joseph A Konstan, Bradley N Miller, David Maltz, Jonathan L Herlocker, Lee R Gordon, and John Riedl. Grouplens: applying collaborative filtering to usenet news. Communications of the ACM, 40(3):77–87, 1997. [6] Yann LeCun, Yoshua Bengio, and Geoffrey H [7] Greg Linden, Brent Smith, and Jeremy York. Amazon. com recommendations: Item-to-item collaborative filtering. IEEE Internet computing, 7(1):76–80, 2003. [8] Hao Wang, SHI Xingjian, and Dit-Yan Yeung. Collaborative recurrent autoencoder: recommend while learning to fill in the blanks. In Advances in Neural Information Processing Systems, pages 415–423, 2016. [9]Mikolov, Tomas, et al. "Distributed representations of words and phrases and their compositionality." *Advances in neural information processing systems*. 2013.