Parts Franck Report

Introduction

With the ever-increasing amount of online information, recommender systems have become a go-to strategy to overcome information overload. Recommender systems facilitate personalization through e-commerce and have grown significantly in terms of public awareness since its rise in the nineties, in parallel to the Web debut. Indeed, may conferences and workshops are exclusively dedicated to this topic, such as the *ACM Conference on Recommender Systems (RecSys)*.

Recommender systems are also very diverse because they may leverage various types of user-preferences and user-requirement data to make the recommendations. The most classical methods in recommender systems include collaborative filtering methods, knowledge-based methods, and content-based methods. They form the fundamental pillars of machine learning based recommender systems.

These methods evolved to fit various scenarios, data domains, and contexts, such as time, location (ie. gps signal) and social media information. Numerous breakthroughs have been found such as the application of Matrix Factorization (MF) and Restricted Boltzmann Machines (RBM), not only for movie recommendation during the 2006 Netflix contest, but also query log mining, news recommendations, and computational advertising.

Conventional methods such as MF are considered state of the art and have been favored over content-based methods, due to dealing better with sparsity, scalability, and predictions. However, they face several challenges such as the cold start problem (new items or users are difficult to predict), very high data sparsity (most of the items are not rated by a user), scalability issues with O(m\*n) complexity in MF, difficulty to boost diversity of the recommended items, and the obstacles to learn latent implicit features.

The last decade has witnessed the Deep Learning (DL) revolution with powerful applications in computer vision, speech recognition, and natural language processing. Currently, it is also revolutionizing the state-of-the-art recommendation architectures by bringing more opportunities to improve performance, which can be generically quantified through RMSE for rating predictions. Indeed, DL approaches have overcome some of the obstacles faced by conventional recommender systems. These approaches can capture the non-trivial and non-linear user-item relationships, and they allow for the representation learning of more complex abstractions as data representations in the higher layers. Further, DL identifies intricate relationships within the data from the abundant sources available at hand such as textual, contextual, and visual information. Thus, the number of published papers has increased exponentially in the recent years. Even RecSys started to organize frequent seminars and workshops on the matter since the year 2016. It is to be noted that DL recommenders also face the following challenges: lack of interpretability, large data requirements, extensive hyperparameter tuning.

As a result, we intend to compare the strengths and limitations of Matrix Factorization (MF), one of the state-of-the-art conventional recommender systems, with DL approaches such as AutoRec and GNN. Other DL algorithms have been studied but we limited our analysis to these two for brevity. First, we will undertake a survey of the literature and explain these methods. Then, we will briefly describe the datasets we used. Third we will describe our experiments with MF, Autorec, and GNC with the goal to optimize prediction rating in user-item context. Finally, we will discuss the results before touching upon the limitations.

Literature Review

Matrix Factorization (MF) is one of the Collaborative Filtering approaches characterized by the reliance on both items and users’ vectors as opposed to Neighborhood based approach [1] that relies on the preferences of the user’s neighbor. Koren et al. have showed The items and users’ vectors are used to infer latent factors of item ratings [2], also referred to as features. MF represents the relation between items and users through that set of latent factors and forms two low rank matrices. Each of them represents the relation between users or the items and the latent features. Finally, the multiplication of these two matrices helps us estimate the user’s future preferences [3]. Mathematically, the rating prediction is: =

For prediction, the loss function is as follows with regularization terms on the users and items vectors:

. The goal of this function to minimize the loss.

It is to be noted there is no common evaluation framework that can be applied to all recommender systems. A variety of measures exist to assess the rating prediction strength of MF techniques and other recommenders in general. There exist three types of metrics to evaluate the power of an algorithm: classification, prediction, and rank accuracy. The classification accuracy assesses the quality of the algorithm to differentiate good items from bad ones. Examples are Precision, Recall, ROC, AUC [4]. The prediction accuracy evaluates the difference between predicted rating and actual rating is the type of metric we chose. Typical prediction metrics are the MAE, normalized MAE, MSE, and RMSE. RMSE has been chosen across our experiments. Rank accuracy measures whether the algorithm sorts the recommended items like the user.

Research in MF and Context Aware MF are abundant. Matrix decomposition has risen as a tool to show latent structures in the data. Some commonly used MF techniques are Singular Value Decomposition [5], PCA, Probabilistic Matrix Factorization [6], and Non-Negative Matrix Factorization [7]. In specific settings, it was showed the SVD could be improved by adding biases to users and items. SVD++ was proposed by Koren et al. to use implicit feedback. It demonstrated a high accuracy but expensive computational costs. Tensor Factorization extends the two-dimensional MF problem into an n-dimensional version by including contextual information [8]. The multi-dimensional matrix is factored into a lower-dimensional one, where each contextual dimension, the item, and the user are represented with a lower dimensional feature vector [9]. Karatzoglou et al suggested a multiverse recommendation system by using CF methods on Tensor Factorization. Additional contextual aware MF models have been studied. Comparative analysis of Boolean Matrix Factorization (BMF) with SVD is undertaken by Akhmatnurov et al [10].

Experiments with the Boolean matrix dot product of binary matrices with or without contextual information with an SVD showed a higher precision for the BMF where the number of user neighbors is not high. Tensor Factorization based on a fuzzy mapping between the latent and contextual factors is proposed by Fang et al [11]. In their paper, movie tags and release time were used as contextual variables and their use led to a higher RMSE and HLU while diminishing the number of iterations by 25%. Finally, we may mention the use of Sparse Linear Method (SLIM) by Zheng et al as Contextual SLIM that incorporates contextual information in a Matrix Factorization approach for Top-N recommendations.

Datasets

The first dataset is the movielens set, it contains three kinds of information: 5-star rating on movies by users; demographic characteristics of users and descriptive characteristics of the movie. We plan to use MovieLens Latest Datasets(Small version for prototyping and Full version for training/testing). The full version of MovieLens Latest dataset has 27,000,000 ratings and 1,100,000 tag applications applied to 58,000 movies by 280,000 users. The dataset contains 6 files: genome-scores.csv, genome-tags.csv, links.csv, movies.csv, ratings.csv and tags.csv. For our experiments on the Movie Lens data, we restricted our investigation on the ratings set and did not add contextual information. The 100k, 1 Million and the 25 Million (later downsized to 2.5 Million) movie lens datasets were used to compare Matrix Factorization performance with Deep Learning approaches such as AutoRec and GNN.

Experiment

EDA Movie Lens

Before deep diving in the prediction problem, we quickly explored the 100k movie lens datasets to better the ratings, items, and users. We focussed on this dataset because it shows movie, user, and ratings trends that are like the trends of the bigger datasets. It is worth noticing that the average age of the users is 42 years old with the youngest being 7 and the oldest being 73. The most popular genres are drama and comedy. Also most of the release occurred in the 1990’s and there is incomplete data at the end of the decade. We also observed that most movies got released on Fridays and week-end which is logical.

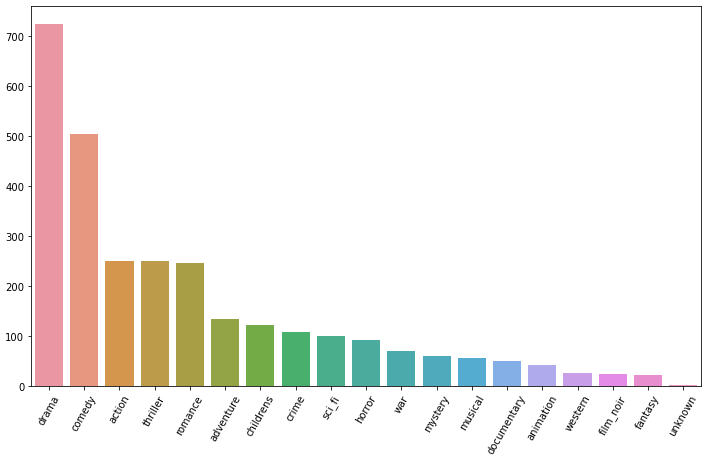
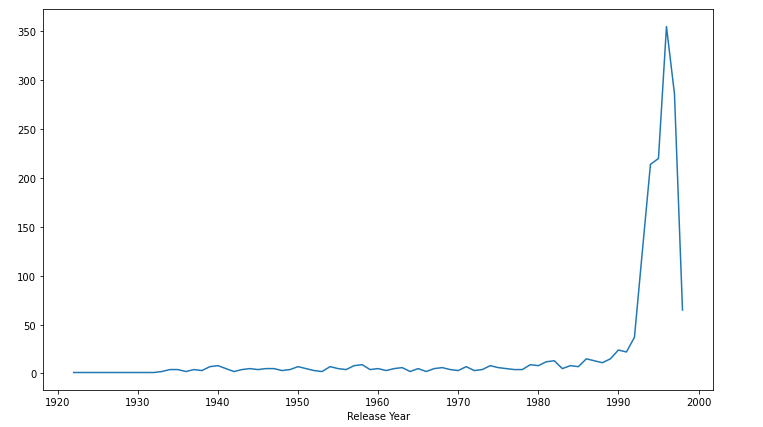


Figure X: Top Movie Genres Figure X: Release year of movies

68 % of the movie raters are male. They are family oriented with 62% of them being either between 30 to 60 or less than 15. There is no significant difference in the ratings given by Male or Female. Our analysis is available in the edaMovieLens100K jupyter notebook under traditionalRSNotebooks.

MF Experiment

We implemented from scratch the Alternating Least Squares MF technique along with an SGD based MF across the 100k and 1 million movie lens, because MF techniques are apparently too computationally expensive on the 2.5 Million dataset – even on GCP. Indeed, fine tuning both MF would take around 1 hour on the 1 million dataset. Across the 2 datasets, every user has rated *at least* 20 movies, leading to a sparsity level of 6.3% in the 100k dataset. We built the ratings matrix with users in the rows and the movies in the columns, the elements in the matrix being the ratings. We split the data between training and validation set by removing 10 ratings per user from the training and placing them in the validation set. After having implemented the proper dot product of the two user and item vectors (for prediction), the ALS loss function with L2 regularization terms on the users/item vectors, and the ALS derivation, we fine tuned the right number of iterations, latent factors k, and the regularization terms through a grid search to minimize the validation RMSE.

For the SGD based MF, we implemented the SGD derivation and an SGD Loss function (like the ALS Loss but with bias terms on the user/item vectors and a global bias term) and optimized the MF SGD model hyperparameters (number of iterations, learning rates, number of latent factors, bias terms, and L2 regularizations) through a grid search to also minimize the validation RMSE.

AutoRec Experiment

Like for the MF techniques, we preprocessed the ratings, movies, and users movie lens datasets, across the 100k, 1 Million, and 25 million restricted to 2.5 million (due to out of memory error) to construct the appropriate rating matrix with the users in the rows and the movies in the columns and the ratings in the matrix. The implementation was done with the tensor flow package with GPU activated in the GCP virtual machine. There is one hidden layer, 1 encoder layer and 1 decoder layer. The training/validation split across the 3 datasets was 75%/25%. We tweaked the following hyperparameters and applied them across the 3 datasets because fine tuning was quite computationally expensive and lengthy : number of layers, optimizer, learning rate, batch size, the activation function, the number of epochs and the number of hidden neurons.

Results/Discussion

We observed that the MF ALS was consistently outperformed by the MF SGD with MF SGD being in the 0.90 in terms of RMSE vs high 2.90 RMSE for the ALS, across the 100k and 1 million movie lens datasets. The MF techniques do not tend to improve their performance with the more data and rather the contrary. The optimal hyperparameters for each technique are available in the summary table. In terms of RMSE, the AutoRec method is at par with MF SGD for the 100k movie lens dataset but beats MF SGD on the 1 million with a RMSE of 0.90 vs 0.95 for MF SGD. Further, the more data, the better the validation RMSE of the AutoRec. Indeed, Autorec was at 1.02 with 100k movie lens, went down to 0.90 with 1 million movie lens dataset to bottom at 0.78 with the 2.5 million dataset. In terms of computational time, we noticed that MF was computationally expensive in terms of fine tuning by taking 1 hour to fine tune (ALS & SGD) on the 1 million dataset. Further, the more the data the more the computational training time for Autorec : 27 minutes on the 100k dataset, 1 hour and 27 minutes on the 1 million dataset and 8 and 13 hours and 39 minutes on the 2.5 million dataset. Finally, we conclude that MF SGD provides the best performance amongst MF but is increasingly computationally intensive with larger datasets while Autorec provides better performance in larger datasets, due to the implicit learning of latent features.

References

[1] Breese, J. S., Heckerman, D., & Kadie, C. (1998). Empirical analysis of predictive algorithms for Collaborative Filtering. In UAI’98 Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence (pp. 43-52). <https://doi.org/10.1111/j.1553-2712.2011.01172.x>

[2] Koren, Y., & Bell, R. (2015). Advances in Collaborative Filtering. Recommender Systems Handbook, Second Edition, 77–118. <https://doi.org/10.1007/978-1-4899-7637-6_3>

[3] Chertov, O., Brun, A., Boyer, A., & Aleksandrova, M. (2015). Comparative analysis of neighborhood-based approache and Matrix Factorization in recommender systems. Eastern-European Journal of Enterprise Technologies, 3(4(75)), 4. https://doi.org/10.15587/1729-4061.2015.43074

[4] Cacheda, F., Carneiro, V., Fernández, D., & Formoso, V. (2011). Comparison of Collaborative Filtering algorithms: Limitations of Current Techniques and Proposals for Scalable, High-Performance Recommender Systems. ACM Transactions on the Web, 5(1), 1-33. <https://doi.org/10.1145/1921591.1921593>

[5] Sarwar, B. M., Karypis, G., Konstan, J. a, & Riedl, J. T. (2000). Application of Dimensionality Reduction in Recommender System - A Case Study. Architecture, 1625, 264-8. <https://doi.org/10.1.1.38.744>

[6] Salakhutdinov, R., & Mnih, A. (2008). Bayesian probabilistic Matrix Factorization using Markov chain Monte Carlo. In Proceedings of the 25th international conference on Machine learning - ICML ’08 (Vol. 25, pp. 880-887). <https://doi.org/10.1145/1390156.1390267>

[7] Cai, D., He, X., Han, J., & Huang, T. S. (2011). Graph regularized Non-negative Matrix Factorization for data representation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 33(8), 1548-1560. <https://doi.org/10.1007/978-3-642-34528-9_1>

[8] Gautam, A., Chaudhary, P., Sindhwani, K., & Bedi, P. (2016). CBCARS: Content boosted Context-aware recommendations using tensor factorization. 2016 International Conference on Advances in Computing, Communications and Informatics, ICACCI 2016, 75-81. <https://doi.org/10.1109/ICACCI.2016.7732028>

[9] Baltrunas, L., Ludwig, B., & Ricci, F. (2011). Matrix Factorization Techniques for Context aware recommendation. In Acm Rs (pp. 301-304). <https://doi.org/10.1145/2043932.2043988>

[10] Akhmatnurov, M., & Ignatov, D. I. (2015). Context-aware recommender system based on Boolean matrix factorisation. In CEUR Workshop Proceedings (Vol. 1466, pp. 99-110).

[11] Fang, Y., & Guo, Y. (2013). A Context-aware Matrix Factorization recommender algorithm. In Proceedings of the IEEE International Conference on Software Engineering and Service Sciences, ICSESS (pp. 914-918). https://doi.org/10.1109/ICSESS.2013.6615454