Ensemble based Movie Recommendation System and Evaluation

Sanket Badhe, Abhishek Bhatt, Siva Harshini Dev Bonam, Soumya Anthareddy
smb535@scarletmail.rutgers.edu,ab2083@scarletmail.rutgers.edu,sdb202@scarletmail.rutgers.edu,sa1607@scarletmail.rutgers.edu
Rutgers University

ABSTRACT

Today we have the ability to collect data from user-item interactions on online platforms. Many such platforms aim to enhance customer engagement, loyalty and purchases by suggesting new items of interest to the user. This is crucial to maximize traffic and revenues for the organization. Content-based recommendations and collaborative filtering are well known approaches to recommend new items to users based on their known ratings of certain items. In practice, there can be many approaches to train a model which take user as input and predict ratings for items that are unrated by that user. New items can then be suggested to the user in the non-increasing order of predicted ratings. However, it is observed that the best results are obtained by combining results from multiple models, rather than using a standalone model.

In this work, we propose an ensemble learning technique that combines ratings generated by collaborative filtering models as well as a content-based model. Our objective is to train a prediction model on the MovieLens Latest Full Dataset of 27,000,000 ratings applied to 58,000 movies by 280,000 users. The model should then be able to recommend movies to users based on predicted ratings such that the recommendations are as relevant, and prediction errors as minimized, as possible.

KEYWORDS

Ensemble Learning, content based recommendations, collaborative filtering, matrix factorization

1 INTRODUCTION

Ensemble learning is an approach that combines different predictive models in order to improve the results obtained by individual models. We apply multiple algorithms to predict ratings, and then build an ensemble from all the results thus obtained. Parallel ensemble is a specific type of ensemble technique, where base learners are generated in parallel (independent of each other), and the prediction error is reduced significantly by averaging the results.

In user-user collaborative filtering, for a given user, we find the set N_u of other users whose ratings are similar to the given user, known as neighbourhood of the given user. Thus, we can find other movies in the set N_u that are rated by users and recommend them to the given user. In item-item collaborative filtering, we similarly find the neighbourhood N_i of a given item (movie).

For implementation, first consider users as rows and movies as columns of a utility matrix. Each entry (i,j) of the utility matrix is the known rating by user i for the movie j. We then factorize this matrix into user to concept similarity matrix U, strength matrix Σ and movie to concept similarity matrix V. Each column in the user-to-concept matrix corresponds to a dimension in the concept space and each entry in U tells us how much a given user corresponds to a given concept. Σ is a diagonal matrix wherein each entry is the weight or strength corresponding to each concept (represented by columns in U). Finally, every entry in V tells how much a given movie corresponds to a concept, with each row representing a dimension in concept space.

Thus, matrix factorization finds the latent factors (concepts) in the utility matrix and maps every movie and every user into a k-dimensional concept space, where k = number of latent factors. Now if we have unknown entries in the utility matrix (say rating of a movie y by user x is not known), we can exploit this property to estimate the rating of a movie y by user x. We only go over known entries of the utility matrix and estimate those entries such that the overall reconstruction error is minimized. We solve this problem using optimization techniques. Finally, we use this model to predict the unknown ratings in the utility matrix.

Content-based recommendations, on the other hand, are based on finding items similar to those preferred (rated) by a given user and recommending them. This is different from collaborative approach in the sense that it does not consider any item or user neighbourhood, i.e. it works irrespective of ratings by other users (or rating of other items).

Dataset: Our training and prediction model is based on the MovieLens dataset [8] collected by GroupLens Research. For this project we used MovieLens Latest Full Dataset (Last updated 9/2018) of 27,000,000 ratings applied to 58,000 movies by 280,000 users. We first trained and tested our model using the 1M dataset (Released 2/2003) of 1 million ratings from 6000 users on 4000 movies, and then scaled the solution to the Latest Full dataset. The results on both the datasets are presented in this work. From each dataset, the data utilized for training and testing the model are the ratings and movies CSV files.

Task: We first split the ratings into a training and testing set. We use the training set to train a rating prediction model based on matrix factorization algorithms. Then we use the model to predict ratings for the users and movies in the testing set. We also use the known ratings in the testing set against the predicted ratings as well as the top N recommendations for each user, to compute evaluation metrics like precision, recall, F-measure, Root Mean Square Error(RMSE) and Mean Absolute Error(MAE). We use these metrics for evaluating the results obtained from multiple

Conference'17, Washington, DC, USA

DOI: 10.1145/nnnnnnn.nnnnnnn

training algorithms like SVD, ALS, NMF, Co-clustering, Slope One, KNN with Baseline, Content Based Approach, and Neural Network Collaborative filtering approach, in order to discover which model(s) gives the most accurate predictions for the same training and testing set. Lastly, we build an ensemble of the results from different models and check for improvements in the evaluation metrics in the aggregated predictions.

2 RELATED WORK

In [15], an overview of recommendation approaches with explanations are presented. [13] discusses ensemble learning, while also highlighting the performance enhancement and limitations of such techniques. In [4], ensembling is used to improve accuracy of recommendations in the context of user-content interactions on the Web. Collaborative filtering and its approaches are discussed in [14]. A Bayesian approach to build a hybrid model for MovieLens is discussed in [2]. The effectiveness of matrix factorization for collaborative filtering was highlighted in the blog post [5], backed the findings of the author's work on the Netflix Prize Problem. Deep learning based approach to discover latent factors in utility matrix is presented in [16].

3 PRELIMINARIES

Exploratory Data Analysis : In the exploratory data analysis we found there are no missing values or redundancies in the data set and hence there was no need for us to clean the data any further to run the recommendation algorithms. Further, by determining the top 10 movies with highest number of ratings, we found that the maximum number of rating is available are for the movie Shawshank Redemption with 97,999 ratings.

In the genre correlations using Pearson correlation method, we found that the standalone attributes with higher correlation above 0.22 are Action-Adventure-Sci-fi-Thriller, Children-Animation-Musical-Fantasy, and Thriller-Crime-Mystery. The least correlated are Comedy-Drama and Comedy-Thriller. Also we can see that the number of movies released per year have increased suddenly between years 1960-1980 and the earliest ratings of all movies are taken from the year 1995. In the end, we saw that Action, Adventure, Crime, Fantasy and War are the five significant genres for the movie ratings obtained from MovieLens dataset.

Data preparation: We read the raw data for user ratings and movies and split it into train and test sets as follows: For each user in the raw data, we keep 80 percent of all the movie ratings by that user in the training set for that user. The remaining 20 percent of all the movie ratings by that user are picked into the testing set for that user. The two partitions thus obtained across all the users form the training and testing data sets for the model.

For the 1M data set, models are trained on 800193 user-movie ratings with 200016 ratings in the test set. For the Latest Full data set, we scale the model by training on 22207659 user-to-movie ratings, using each algorithm. Predictions are then made and evaluated against the 5545785 user-to-movie ratings in the testing set, for each algorithm as well as the averaged ensemble.

4 PROBLEM FORMALIZATION

The goal of this work is to build a movie recommendation system and evaluate the accuracy of the recommendations given by it. The inputs to the system include two positive integers : userId and n. The system returns a list of top n movie recommendations for the given user as well as evaluation metrics using which we quantify the prediction errors. Our aim is to get the optimal values for the metrics that measure the two components of the output : the degree to which relevant results are obtained in the top n recommendations, and the degree of disparity between predicted and true ratings. We compute these metrics mathematically and evaluate recommendation models using them as described below.

Precision : Precision is defined as Precision = TP/(TP+FP), where TP = number of predicted results that were also expected results FP = number of predicted results that were not expected

Recall : Recall is defined as Recall = TP/(TP+FN), where TP = number of predicted results that were also expected results FN = number of expected results that were not predicted

F-Measure : It is observed[6] that there is usually an inverse relationship between precision and recall. Both of the scenarios, low recall and high precision, or vice versa are unacceptable in practical use cases. The optimal model is thus the one where we maximize for one metric, such that the other metric is maintained above a certain threshold.

F Measure or F Score is defined as the harmonic mean (average) of precision and recall. Desired value of F Score is obtained when there is some sort of balance between precision and recall in the system, as opposed to low F Score, when one measure is improved at the expense of the other.

F-Measure = 2*(Recall * Precision) / (Recall + Precision)

MAE : Mean Absolute Error(MAE) measures the average magnitude of the errors in a set of predictions, without considering their direction. In other words, it is the average over the test sample of the absolute differences between prediction and actual observation where all individual differences have equal weight.

RMSE : Root Mean Square Error(RMSE) is a quadratic scoring rule that also measures the average magnitude of the error. It is the square root of the average of squared differences between prediction and actual observation

Both MAE and RMSE express average model prediction error in units of the variable of interest. Both metrics can range from 0 to and are indifferent to the direction of errors. They are negatively-oriented scores, which means lower values are better[18].

5 THE PROPOSED MODEL

Ensemble Model: The ensemble model comprises of the mean ratings predicted by each of the individual models discussed below. The evaluation metrics of the ensemble are demonstrated under results versus each individual model for both 1M and full Latest datasets of MovieLens.

Ensemble technique balanced the bias variance trade-off. Aggregating multiple predictions also filters out the noise in the standalone model results, thereby reducing the variance. Lastly, this technique avoids any possibility of over-fitting. As a result, the averaged predictions tend to have the best accuracy.

Lastly, We can combine different types of algorithms using ensemble techniques such as Content based approach, collaborative filtering, learning to rank methods etc.

Neural Network: Neural Network Collaborative filtering tries to learn user-movie interaction through the deep neural network and generalize matrix factorization. In this framework, we used two embedding layers to represent the Movies and Users. For the movie embedding layer, we kept latent factor as 8 and For users embedding layer, we kept latent factor as 5. In the next step, we flattened the movie layers as well as the user layer. Then we used the concatenate layer to merge both flattened layers. Lastly, we used multiple fully connected dense layers. We also added multiple dropout layers in our network to avoid over-fitting. We used the rating matrix as the target variable.

Model: "model_1"

Layer (type)	Output Shape	Param #	Connected to
Movie (InputLayer)	(None, 1)	0	
User (InputLayer)	(None, 1)	0	
Movie-Embedding (Embedding)	(None, 1, 8)	431120	Movie[0][0]
User-Embedding (Embedding)	(None, 1, 5)	1416145	User[0][0]
FlattenMovies (Flatten)	(None, 8)	0	Movie-Embedding[0][0]
FlattenUsers (Flatten)	(None, 5)	0	User-Embedding[0][0]
dropout_1 (Dropout)	(None, 8)	0	FlattenMovies[0][0]
dropout_2 (Dropout)	(None, 5)	0	FlattenUsers[0][0]
concatenate_1 (Concatenate)	(None, 13)	0	dropout_1[0][0] dropout_2[0][0]
FullyConnected-1 (Dense)	(None, 100)	1400	concatenate_1[0][0]
FullyConnected-2 (Dense)	(None, 50)	5050	FullyConnected-1[0][0]
FullyConnected-3 (Dense)	(None, 20)	1020	FullyConnected-2[0][0]
Activation (Dense)	(None, 1)	21	FullyConnected-3[0][0]

Total params: 1,854,756
Trainable params: 1,854,756
Non-trainable params: 0

Content Based: The content-based recommendation approach recommends movies to the users according to their preferences and tastes by considering the user's profile. In our approach, we used a list of genres for a movie as content. We created features and vectorized each movie using TF-IDF. Next, we find cosine similarity between movies the user has rated and movies we need to be recommended. For predicting rating for the movie i (movie_i), we computed the weighted average rating with all the movies that the user has rated(movie_u) and have positive cosine similarity with movie_i. Where weight corresponds to cosine similarity between movie_u and movie_i. As we are only considering the positive cosine similarity while calculating the weighted average, if there is no movie in movie_u with positive cosine similarity for movie_i, then we just take mean of movie_u as the rating for movie_i.

SVD: Singular Value Decomposition (SVD)is a matrix decomposition technique which can be used for dimentionality reduction, noise reduction and also compression. SVD is more popular because any real matrix can be decomposed making this technique more stable than even eigen decomposition. In this procedure a real matrix is decomposed into three matrices, say U, Sigma, V^T, where U and V are composed of orthonormal columns and Sigma is a diagonal matrix with singular values, which help us deduce the most significant correlated properties.

NMF: Non-negative Matrix Factorization (NMF)[19] is a dimensionality reduction technique based on decomposition by parts. It takes the n x m utility matrix V with non-negative ratings, where n = total number of users and m = total number of movies, and decomposes it into two non-negative matrices W (n x k) and H (k x m). This is done by solving the optimization problem, minimize $\|V - WH\|$ such that W≥0 and H≥0. A method to solve this optimization using Gradient Descent with Multiplicative Update is discussed in [9].

NMF has an inherent clustering property[3], it automatically clusters the columns of input matrix V. Furthermore, the computed matrix H gives the cluster membership, i.e., if $H_{kj} \geq H_{ij}$ for all i k, this suggests that the input data v_j belongs to k^{th} cluster. The matrix W gives the cluster centroids, i.e., the k^{th} column gives the cluster centroid of k^{th} cluster. NMF therefore, discovers latent factors in the utility matrix and maps each movie and each user to a k-dimensional concept space.

SlopeOne: Slope One for collaborative filtering was developed by Lemire and Maclachlan[10]. Linear regression f(x)=ax+b can be used to predict the ratings on one item based on the ratings on another item [17]. However, severe overfitting issues may arise in this method if pairs of items are selected such that few users have rated both.

[20]A better alternative is using a simpler predictor such as f(x)=x+b (called Slope One), that is experimentally found to give better performance than linear regression, with smaller number of regressors. The free parameter b is the average difference between the two items' ratings. The modified predictor is also efficient in terms of memory requirements and latency.

KNN with Baseline: K-Nearest Neighbours algorithm [1] finds the unknown ratings of an item using k of the ratings in its neighbourhood. KNN computes similarity of the given item, with other items with known ratings in its neighbourhood, using the Centered Cosine Distance (Pearson Correlation) metric. KNN with baseline [12] also includes a base rating for each user-item pair besides the similarity metric. Including baseline estimate models the biases in the prediction thus enhancing the accuracy and reliability of the predictions obtained.

CoClustering: Co clustering or biclustering [7] allows simultaneous clustering along the rows and columns of the utility matrix. Users and items are assigned to some clusters C_u, C_i, and also some co-clusters C_{ui}[11].

$r_{ui} = \overline{C_{ui}} + (\mu_u - \overline{C_u}) + (\mu_i - \overline{C_i})$

where $\overline{C_u i}$ is the average rating of co-cluster C_{ui}, $\overline{C_u}$ is the average rating of u's cluster, and $\overline{C_i}$ is the average rating of i's cluster. If the user is unknown, the prediction is $r^u i = i$. If the item is unknown, the prediction is $r^u i = u$. If both the user and the item are unknown, the prediction is $r^u i =$.

The clustering itself is done through optimization techniques such as k-means.

ALS: Alternating Least Squares(ALS) is also a matrix factorization algorithm. It is simple and solves scalability and sparseness of the ratings data. ALS runs in a parallel way, minimizing two loss functions alternatively. It first fixes the user matrix and runs gradient descent with item matrix, then it fixes the item matrix and runs gradient descent with user matrix. This way it is useful for larges-scale collaborative filtering problems[21].

Table 1: Experiment results on 1 Million dataset

method	RMSE	MAE	Precision	Recall	FMeasure
NeuralNet	0.95	0.71	0.67	0.67	0.67
contentBased	0.94	0.71	0.66	0.66	0.66
SVD	0.87	0.68	0.68	0.68	0.68
NMF	0.91	0.72	0.66	0.66	0.66
CoClustering	0.91	0.71	0.67	0.67	0.67
ALS	1.25	1.16	0.62	0.62	0.62
KNN	0.89	0.70	0.67	0.67	0.67
Ensemble	0.87	0.68	0.68	0.68	0.68

Table 2: Experiment results on 27 Million data

method	RMSE	MAE	Precision	Recall	FMeasure
NeuralNet	0.90	0.66	0.83	0.83	0.83
SVD	0.78	0.59	0.85	0.85	0.85
NMF	0.87	0.67	0.82	0.82	0.82
CoClustering	0.89	0.69	0.83	0.83	0.83
ALS	1.29	1.01	0.77	0.77	0.77
Ensemble	0.80	0.61	0.84	0.84	0.84

6 EXPERIMENTS

The table 1 presents metrics computed for various individual models as well the ensemble when trained and tested for the 1M MovieLens dataset of approximately 1 million records.

Table 2 shows the metrics obtained when the models were scaled for the Full Latest MovieLens dataset of approximately 27 million records. Due to computing power limitation and high space complexity of some algorithms presented for 1M, only few of those models were included for analysis on 27 million rating dataset.

Because of large size of dataset most of model was not difficult to train on our local machine. We used google colab platform for the training our algorithms.We ran our neural network model for 15 epochs. we build our model with Keras(Tensorflow backend). For training neural network we used adam optimizer and mean absolute error as loss function. For training ALS we used pyspark's MLLib library of iteration as 20 and for training SVD, KNN, Slopeone, Coclustering, NMF we used surprise library.

7 CONCLUSIONS AND FUTURE WORK

We can see for 1M moivelens dataset, we got best evaluation metrics results by taking ensemble of the all the individual model. For 27M movielens dataset though our evaluation metrics is not better than SVM but after Ensemble Rating outperform all the other method except SVM.

In conclusion, ensemble techniques balanced the bias variance trade-off in ensemble learning and provide better result then base learner.

In future work, we can improve our model by using weighted average of the rating rather than taking simple average. Our result might improve more if we incorporate learning to rank based approached in base model.

REFERENCES

[1] N. S. Altman. 1992. An Introduction to Kernel and Nearest-Neighbor Nonparametric Regression. *The American Statistician* 46, 3 (1992), 175–185. http://www.jstor.org/stable/2685209

[2] Luis M. [de Campos], Juan M. Fernández-Luna, Juan F. Huete, and Miguel A. Rueda-Morales. 2010. Combining content-based and collaborative recommendations: A hybrid approach based on Bayesian networks. *International Journal of Approximate Reasoning* 51, 7 (2010), 785 – 799. DOI: http://dx.doi.org/https://doi.org/10.1016/j.ijar.2010.04.001

[3] Chris Ding, Xiaofeng He, and Horst D. Simon. 2005. On the equivalence of nonnegative matrix factorization and spectral clustering. In *in SIAM International Conference on Data Mining*.

[4] Arthur Fortes and Marcelo Manzato. 2014. Ensemble Learning in Recommender Systems: Combining Multiple User Interactions for Ranking Personalization. *WebMedia 2014 - Proceedings of the 20th Brazilian Symposium on Multimedia and the Web* (11 2014), 47–54. DOI: http://dx.doi.org/10.1145/2664551.2664556

[5] Simon Funk. 2006. Netflix Update: Try This at Home. (2006). https://sifter.org/~simon/journal/20061211.html

[6] Salma Ghoneim. 2019. Accuracy, Recall, Precision, F-Score Specificity, which to optimize on? (2019). https://towardsdatascience.com/accuracy-recall-precision-f-score-specificity-which-to-optimize-on-867d3f11124

[7] Gérard Govaert and Mohamed Nadif. 2008. Block clustering with Bernoulli mixture models: Comparison of different approaches. *Computational Statistics & Data Analysis* 52, 6 (February 2008), 3233–3245. https://ideas.repec.org/a/eee/csdana/v52y2008i6p3233-3245.html

[8] GroupLens. 2019. MovieLens. (2019). https://grouplens.org/datasets/movielens/

[9] Daniel D. Lee and H. Sebastian Seung. 2001. Algorithms for Non-negative Matrix Factorization. In *Advances in Neural Information Processing Systems 13*, T. K. Leen, T. G. Dietterich, and V. Tresp (Eds.). MIT Press, 556–562. http://papers.nips.cc/paper/1861-algorithms-for-non-negative-matrix-factorization.pdf

[10] Daniel Lemire and Anna Maclachlan. 2007. Slope One Predictors for Online Rating-Based Collaborative Filtering. (2007). arXiv:cs.DB/cs/0702144

[11] NicolasHug. 2020. Co-clustering. (2020). https://surprise.readthedocs.io/en/stable/co_clustering.html

[12] NicolasHug. 2020. k-NN inspired algorithms. (2020). https://surprise.readthedocs.io/en/stable/knn_inspired.html#surprise.prediction_algorithms.knns.KNNBaseline

[13] D. Opitz and R. Maclin. 1999. Popular Ensemble Methods: An Empirical Study. *Journal of Artificial Intelligence Research* 11 (Aug 1999), 169–198. DOI: http://dx.doi.org/10.1613/jair.614

[14] Xiaoyuan Su and Taghi Khoshgoftaar. 2009. A Survey of Collaborative Filtering Techniques. *Adv. Artificial Intellegence* 2009 (10 2009). DOI: http://dx.doi.org/10.1155/2009/421425

[15] Loren Terveen and Will Hill. 2001. Beyond Recommender Systems: Helping People Help Each Other. In *HCI in the New Millennium*. Addison-Wesley, 487–509.

[16] Moshe Unger, Ariel Bar, Bracha Shapira, and Lior Rokach. 2016. Towards latent context-aware recommendation systems. *Knowledge-Based Systems* 104 (2016), 165 – 178. DOI: http://dx.doi.org/https://doi.org/10.1016/j.knosys.2016.04.020

[17] Slobodan Vucetic and Zoran Obradovic. 2005. Collaborative Filtering Using a Regression-Based Approach. *Knowl. Inf. Syst.* 7 (02 2005), 1–22. DOI: http://dx.doi.org/10.1007/s10115-003-0123-8

[18] Janet Wesner. 2016. MAE and RMSE — Which Metric is Better? (2016). https://medium.com/human-in-a-machine-world/mae-and-rmse-which-metric-is-better-e60ac3bde13d

[19] Wikipedia. 2020. Non-negative matrix factorization. (2020). https://en.wikipedia.org/wiki/Non-negative_matrix_factorization#cite_note-lee2001algorithms-14

[20] Wikipedia. 2020. Slope One. (2020). https://en.wikipedia.org/wiki/Slope_One

[21] Robert Schreiber Yunhong Zhou, Dennis Wilkinson and Rong Pan. 2009. Large-scale Parallel Collaborative Filtering for the Netflix Prize. (2009). https://endymecy.gitbooks.io/spark-ml-source-analysis/content/%E6%8E%A8%E8%8D%90/papers/Large-scale%20Parallel%20Collaborative%20Filtering%20the%20Netflix%20Prize.pdf