Music recommendation system final report

Yiqun Gan, Ziheng Yu, Tianxing Wang

project description

For this project, we are going to build a music recommendation with KKBox dataset with the hybrid recommendation method. For recommendation system parts, we separately use three methods, user-based, item-based, and (SVD) model based to get independent recommendation systems first, then we adopt the optimized hybrid weighted method to combine three of them to recommend. In addition, we apply machine learning to regard the recommendation as a classifier and utilize the adaboost and catboost to obtain a strong classifier. In this process, we apply many machine learning methods, such as decision tree, random forest, and XGBoost. Finally, we use different evaluation methods to evaluate the performance of our models and both methods could yield good performance which are significant promotions for the original single method.

Data description

Our Dataset is coming from Kaggle competition - KKBox's music Recommendation Challenge. The dataset includes information of the first observable listening event for each unique user-song pair within a specific time duration. If there are recurring listening event(s) triggered within a month after the user's very first observable listening event, the target is marked 1, and 0 otherwise. The data has 30,755 users, 359,966 songs, 7,377,418 records. There are 11 categorical features and 6 numerical features.

New and innovative

Music platforms usually didn't provide a rating system for users like movies or books. Predicting whether a song is favored by a user is mainly decided by their repetitive listening behavior. It is a classification problem as we'd like to predict if the user likes a song or not. KKBOX, Asia's leading music streaming service, currently uses a collaborative filtering based algorithm in their recommendation system. We are interested in exploring whether hybrid collaborative filtering models or other machine learning of classification models could lead to better results in music recommendation.

Pose and motivate hypotheses

Initially, we plan to build three collaborative filtering models. They are user based model, item based model and a singular value decomposition model respectively. The user based model is based on finding similar users by calculating similarity of their favorite songs. The item based model is based on finding similar songs by calculating similarity of popularity between users. The third model will be built by singular value decomposition. We assume these models are all biased, a hybrid model can have better performance and help improve the predictive accuracy.

For the hybrid recommendation system, we need to make sure the weighted sum from three recommendation systems is as close as possible to the ground truth labels and thus we could get a better performance hybrid recommender than any single recommendation model. So we just assume that we could obtain an optimized combination from the formula below[1][2].

$$\tilde{R}_{ui} = \sum_{f}^{c} \delta_f R_{ui}^{(f)} \quad (1)$$

In this formula, σ represents the weight for every single recommendation result R_{ui}. In order to get a optimized combinations and values for weights, we need to define the error function

$$E(\delta) = \sum_{u} \sum_{i} \left(R_{ui} - \delta_1 R_{ui}^{(2)} - \ldots - \delta_c R_{ui}^{(c)} \right)^2 \quad (2)$$

$$s.t. \sum_{f} \delta_f = 1 \quad (3)$$

so our problem becomes how to minimize the equation. (sum of weights is equal to 1)

For minimization,We could build a Lagrange function as

$$L = \frac{1}{2} E(\delta) + \lambda \left(\sum_{f} \delta_f - 1 \right) \quad (4)$$

And we let $\frac{\vartheta L}{\vartheta \delta_f} = 0$, then

$$\sum_{u} \sum_{i} \left(\delta_1 R_{ui}^{(1)} + \delta_2 R_{ui}^{(2)} + \ldots + \delta_c R_{ui}^{c} \right) R_{ui}^{(f)} + \lambda = \sum_{u} \sum_{i} R_{ui} R_{ui}^{(f)} \quad (5)$$

We could get the matrix equation

$$\begin{bmatrix} \sum_{u} \sum_{i} R_{ui}^{(1)} R_{ui}^{(1)} & . & . & \sum_{u} \sum_{i} R_{ui}^{(1)} R_{ui}^{(c)} & 1 \\ . & . & . & . & 1 \\ . & . & . & . & 1 \\ . & . & . & . & 1 \\ \sum_{u} \sum_{i} R_{ui}^{(c)} R_{ui}^{(1)} & . & . & \sum_{u} \sum_{i} R_{ui}^{(c)} R_{ui}^{(c)} & 1 \\ 1 & . & . & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} \delta_1 \\ . \\ . \\ . \\ \delta_c \\ \lambda \end{bmatrix} = \begin{bmatrix} \sum_{u} \sum_{i} R_{ui} R_{ui}^{(1)} \\ . \\ . \\ . \\ \sum_{u} \sum_{i} R_{ui} R_{ui}^{(c)} \\ 1 \end{bmatrix} \quad (6)$$

In this matrix equation, X is the weight vector and our target. So we could let

$$X = \alpha \cdot A^{-1} \quad (7)$$

This is the optimized weights parameters that we need.

The collaborative filtering model doesn’t take metadata and extra information into consideration while machine learning models use label encoding to make use of categorical features. We assume that machine learning of classification models like decision tree, random forest, XGBoost could have better results than a single collaborative filtering model. The reason we use the tree base model is because the random forest model randomly selects the features and random sampling and XGBoost builds the tree model sequentially and improves based on the previous tree's residual. All tree models can provide feature importance.Also a hybrid machine learning model using AdaBoosting may help improve the accuracy of the machine learning model.

Initialize the observation weights w_i = 1/N, i = 1,2,...,N. We train the data using a classifier, then we calculate weighted error rate which gives higher weight to wrongly classified points. In the next iteration, we train another model with data of updated weights[6].

For m = 1 to M (number of models):

a. Fit a classifier $G_m(x)$ to the training data using weights w_i
b. Compute $err_m = \frac{\sum_{i=1}^{N} w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i}$
c. Compute $\alpha_m = \log((1 - err_m) / err_m)$
d. Set $w_i \leftarrow w_i \cdot exp[\alpha_m \cdot I(y_i \neq G_m(x_i))],\ i = 1, 2, \ldots, N$

Output $G(x) = sign\left[\sum_{m=1}^{M} \alpha_m G_m(x)\right]$

Literature Review:Historical Implementation

In a typical music recommendation system, the collaborative filtering method is preferred by using user behavior and that of similar users. One perfect example is Spotify. The idea behind is that the system finds similar songs that appear on the users playlists and the other billions of users playlists, and then based on the user tastes, the system would recommend the similar songs that the user has not listened to.

Besides, Spotify also uses NLP technique to identify the co-location of individual terms and uses this to predict the meaning of phrases. This serves big differences when we deal with music as a language that speaks in particular words[3].

Another technique serves more on the new song purposes. Instead, Spotify uses the kind of neural network that is employed by search engines to understand the contents of audio images across a range of characteristics, including key, tempo, and even loudness.

Spotify uses collaborative filtering methods associated with NLP and Audio models to quickly identify similar music to the one similar to users' likes and make recommendations[3].

In addition, the simple & fast LGBM method, a gradient boosting framework that uses tree based learning algorithms, is usually adopted in the problem of music recommendation systems and could make a good recommendation with the score of 0.6685 in the kaggle competition[4][5].

Result

Evaluation approach

For the evaluation method, we adopt the AUC and ROC score as our evaluation index. ROC, Receiver Operator Characteristic curve, is an evaluation metric for binary classification problems. AUC, Area Under the Curve, is the measure of the ability of a classifier to distinguish between classes and is used as a summary of the ROC curve.

For building the AUC curve, we need to first build the confusion matrix as Figure 1. We could find that the true positive rate is TP/(TP+FN), and the false positive rate is FP/(TN + FP).We let the true positive rate as Y column and false positive rate as X column, we then could get the ROC curve as Figure 2. The area of the ROC curve is AUC score and that is what we want. Commonly, the higher AUC score is, the better the performance of the model is, and the lower false positive rate is preferred by the music recommendation.

Hybrid method and collaborative filtering

After training three different recommendation systems (user-based, item-based, model-based) and adopting the optimized weight hybrid method above. We need to bring their training labels into the matrix equation above and finally obtain optimized weight for each recommendation system and check the hybrid optimized model performance in the test set with ground true labels.

In this process, we utilize the cross validation method and ROC and AUC score as our evaluation index. In this phase, we take 10-fold cross validation, splitting the data into 10 sets and iterate the test set as one of it, remaining as a train set. Finally, we could get an average performance of the scores 0.73, which is a significant promotion compared with the single recommendation system (user-based 0.70, item-based 0.64, model-based 0.63). And it also proves that the methods and assumptions that we have are correct about the optimized hybrid weighted model as shown in Figure 3 and Table 1.

Machine learning

Our machine learning methodology involves 3 parts including filling missing variables, feature engineering, and modeling. We fill the missing variable by using the high frequent record for categorical variable and mean value for numerical variable. Then we encode the categorical variable using the number of appearance values and create additional several time features, such as splitting user account registration time into days, months, and years. We use 3 different modeling approaches including decision tree, random forest, and XGBoost. This gives us accuracy results between 0.66 and 0.68. Then we use the Voting Classifier to aggregate the findings of each classifier passed into the classifier to predict the output class based on the highest majority of voting. The Adaboost ensemble three models and predict an output based on their highest probability of chosen class as the output. This improves our accuracy results from each individual model up to 0.72. This seems to consolidate our machine learning hypothesis, which combines models that perform better than each individual one as shown in Figure 4 and Table 2.

CatBoost

CatBoost[7] is an open source library for gradient boosting on decision trees. It can make full use of categorical features. No label encoding or one hot encoding needed. As most features of our dataset are categorical, CatBoost is a good candidate for building music recommendation systems. After training a model using CatBoost, we get an AUC score of 0.80.

Conclusion, lessons learned, future work

Conclusion

Collaborative filtering has no complex data preprocessing, it is easy to use. However, metadata or extra information is not taken into consideration, which could be improved in the future. Also parameter tuning can be a good way to improve the model.

Machine learning models are mainly focused on data preprocessing, the data size and feature can be enlarged. It is computational consuming, and the hybrid method is a challenge for large datasets. CatBoost is a good approach for classification problems.

Optimization methods can be used to combine different models without re-training the model while adaboost needs to train the model again in each iteration, large computational power is required.

Lessons learned

From this project, we learned lots of machine learning models, like XGBoost, random forest, and catboost, each of which is a representative and classical method.

For the project, we also learned how to apply the optimized hybrid methods for multiple recommendation systems to obtain a better result than any single recommender.

Besides the data exploratory methods are also important for us to learn about features of the dataset and we could apply the evaluation method for specific models with their characteristics.

Future work

It could be hard to process super massive data like million levels, and we could build distributed data server farms to solve the problem with flink or spark. More features could be used to improve the machine learning and recommendation system performance. Even though our approach improves the accuracy in a good way, the cold start problem is still an issue. In the future, we could implement audio related data or enrich the song genre characteristics.

References

[1] The optimization of weights in weighted hybrid recommendation algorithm Wenfu Lin;Ying Li;Shuang Feng;Yongbin Wang 2014 IEEE/ACIS 13th International Conference on Computer and Information Science (ICIS).
[2] Recommender Systems: The Textbook Aggarwal, Charu C Cham: Springer International Publishing AG; 2016
[3]https://towardsdatascience.com/how-spotify-recommends-your-new-favorite-artist-8c1850512af0#:~:text=Well%2C%20Spotify's%20recommender%20system%20provides,it%20deems%20%E2%80%9Csimilar%E2%80%9D%20users.
[4] https://www.kaggle.com/kamilkk/simple-fast-lgbm-0-6685/comments
[5] https://github.com/microsoft/LightGBM
[6] https://web.stanford.edu/~hastie/ElemStatLearn/
[7] https://www.kaggle.com/prashant111/catboost-classifier-in-python

Appendix

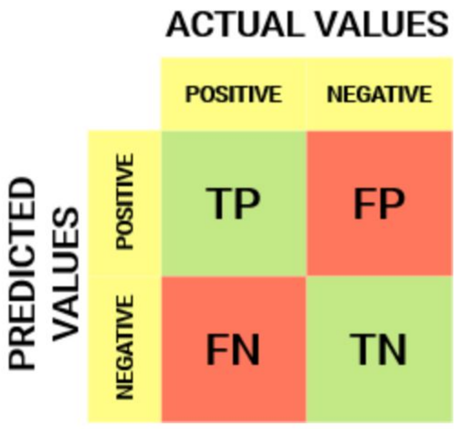

Figure 1. Confusion Matrix

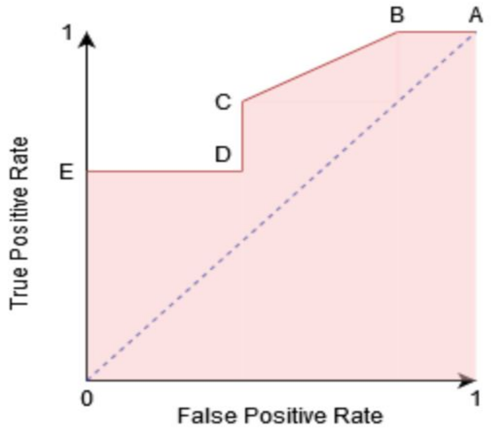

Figure 2. AUC Curve

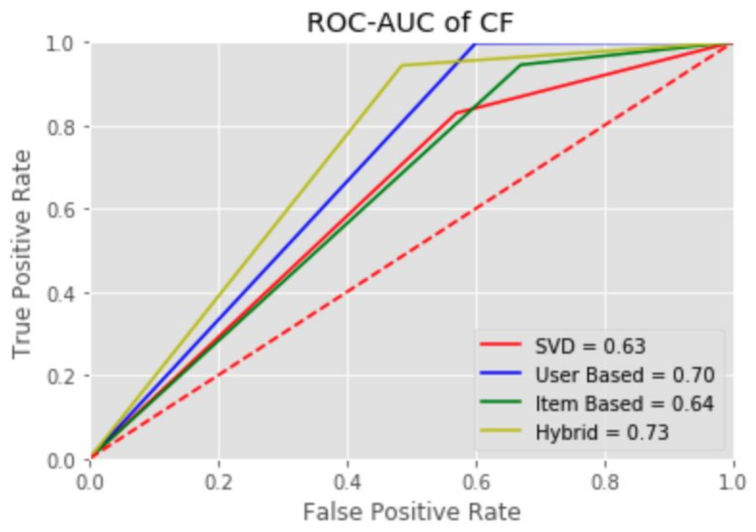

Figure 3. Recommender System Hybrid AUC

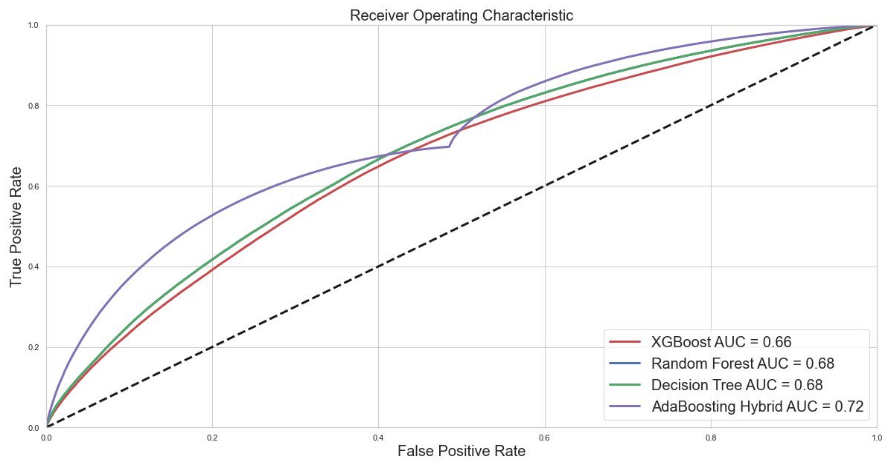

Figure 4. Machine Learning AUC

Model	ROC-AUC
SVD	0.63
User Based	0.70
Item Based	0.64
Hybrid(optimization)	0.73

Table 1. Hybrid Collaborative Filtering AUC Value

Model	ROC-AUC
Decision Tree	0.66
Random Forest	0.68
XGBoost	0.68
Hybrid(AdaBoost)	0.72

Table 2. Machine Learning AUC Result