Incorporating Field-aware Deep Embedding Networks and Gradient Boosting Decision Trees for Music Recommendation

The 1st Place Music Recommender System at WSDM Cup 2018

Bing Bai
Tsinghua National Laboratory for Information Science and Technology
Department of Automation
Tsinghua University
Beijing 100084, China
bb13@mails.tsinghua.edu.cn

Yushun Fan*
Tsinghua National Laboratory for Information Science and Technology
Department of Automation
Tsinghua University
Beijing 100084, China
fanyus@tsinghua.edu.cn

ABSTRACT

On-line music streaming services, like KKBOX, is bringing great convenience for users to get access to all kinds of music, while the hierarchical data describing the songs and the random interests of users still make it a challenge for algorithms to make accurate recommendations, especially without enough historical data. In the WSDM - KKBOX's Music Recommendation Challenge, WSDM and KKBOX were challenging people to predict the chances of a user listening to a song repetitively, under the condition that many users/songs are cold-started. In this paper, we describe our solution to the task. Our solution comprised an ensemble of different models, including Field-aware Deep Embedding Networks and Gradient Boosting Decision Trees. We achieved an AUC score of 0.74787 on the Private Leaderboard, and finished the first place in the competition.

Keywords

Music recommendation, recommender system, deep learning, field-aware, GBDT, WSDM Cup

1. INTRODUCTION

With the rapid development of on-line music streaming services, people can get access to a great many songs of all kinds of genres in a convenient manner. The same person could listen to the Beatles, Vivaldi, and Lady Gaga on their morning commute, which was hardly conceivable in the past. By mining the massive listening records, music streaming service providers can make personalized recommendations to users, thus relieve the information overload.

However, the trends in music are always evolving. New songs and new artists emerge every week, which puts a great challenge to recommendation algorithms. Without enough historical data, how would an algorithm know if listeners will like a new song or a new artist? And, how would it know what songs to recommend to brand new users? WSDM has challenged competitors to build a better music recommendation system with a donated dataset from KKBOX[1], which is Asia's leading music streaming service, holding the world's most comprehensive Asia-Pop music library with over 30 million tracks.

The dataset contains 7,377,418 records for the first observable listening events in the training set, 2,556,790 records in the testing set, and the goal is to predict the chances of a user listening to a song repetitively within a time window after the first observable listening event was triggered. The dataset involves 34,403 distinct users where 10.60% of the users do not appear in the training set, and 419,839 distinct songs where 14.26% of them do not appear in the training set. The metadata of users and songs is also provided. Based on the observation from competitors, the training set covers a range of time from mid-August 2016 to mid-January 2017, and the testing set is from mid-January 2017 to late February 2017.

Based on our understanding, the problem has the following properties:

- **Missing not at random**. Some user-song pairs can be observed and some not. If a user-song pair appears in the dataset, we can tell that the user has listened to the song at least once, which means that the user is more or less interested in the song. As a result, the co-occurrence of users and songs contains rich information about user preference and song characteristics [18].

- **Time-sensitive**. Since the dataset covers a long range of time, the pattern might evolve a lot from the beginning to the end. This makes the training/testing data follow more or less different distributions, and is important for the validation strategy and feature engineering.

Based on these properties, we propose to incorporate Field-aware Deep Embedding Networks (FDEN) and Gradient Boosting Decision Trees (GBDT) to make recommendations. GBDTs are greedy methods and can fit the pattern in the head flow well, while FDENs explore the broad combinations of features, thus are better at finding the pattern hiding in the flow of the long-tail. As a result, the ensemble based on the predictions given by FDENs and GBDTs can give a significant boost to each of them. Finally we were able

*This is the corresponding author.
[1]https://www.kkbox.com/

to get 0.74787 on the Private Leaderboard, and finished the first place in the competition.

The rest of the paper is organized as follows: Section 2 summarizes some related work. Section 3 describes the overall approach and some important feature engineering methods. Section 4 introduces the detailed structure of the FDEN and Section 5 introduces the GBDT model. Section 6 provides an overview of the evaluation results, and Section 7 draws the conclusion.

2. RELATED WORK

Music recommender systems have been an active research topic for many years [16]. Cheng et al. [4] proposed a location-aware topic model mining the common features of songs that are suitable for a venue type to make location-aware music recommendations. Rosa et al. [14] extracted users' sentiments from social networks and focused on their relationship with personalities to infer music taste and preferences. Liebman et al. [11] proposed a novel reinforcement-learning framework for music recommendation that does not recommend songs individually but rather playlists. Schedl introduced the LFM-1b dataset of more than one billion music listening events created by more than 120,000 users of Last.fm for music retrieve and recommendation in [15].

Beyond the field of music recommendation, applying deep learning to make recommendations also attracts a lot of researchers' attention. Cheng et al. [3] jointly trained wide linear models and deep neural networks to combine the benefits of memorization and generalization for recommender systems. Wang et al. [20] tightly coupled deep neural networks and collaborative filtering to make more effective recommendations. Zhou et al. [22] proposed a Deep Interest Network to represent users' diverse interests with an interest distribution and make better predictions on the click through rate.

In this paper, we introduce a tailored structure of neural networks called Field-aware Deep Embedding Network for music recommendation. It can serve as a significant complement to GBDTs and boost the performance by ensembles.

3. OVERALL FRAMEWORK AND FEATURE ENGINEERING

In this section, first the overall framework of our approach is presented, and then some important feature engineering methods are introduced.

3.1 Overall Framework

As described in the Introduction, we use an ensemble of FDENs and GBDTs. Since the data is time-sensitive, stacking ensemble [5] and blending ensemble [21] is not working well, so we use a relatively simple form of ensemble. The final score is a weighted average of predictions from the two kinds of models, i.e.,

$$score_{\text{ensemble}} = 0.4 * score_{\text{FDEN}} + 0.6 * score_{\text{GBDT}}, \quad (1)$$

where the weights are tuned based on scores from the Public Leaderboard.

The predictions of FDENs (i.e., $score_{\text{FDEN}}$) are from a bagging ensemble using the arithmetic mean of many networks, each of which has slight differences on hyper-parameters, including the forms of the activations, the $\ell 2$ regularizations, the number of nodes in hidden layers and so on. Besides, we also tried training the models with slightly different feature sets to get more diversity for the ensemble.

As for GBDTs, they are relatively insensitive to the change of hyperparameters, so we only use the ensemble using weighted mean of GBDTs with slightly different feature sets. The weights are tuned based on the Public Leaderboard.

Due to the properties of the models, FDENs and GBDTs give quite uncorrelated predictions. GBDTs are greedy methods, and always captures the pattern in the head flow first, and then the trees behind can finally fit the records in the tail flow. While FDENs, in contrast, explore the broad combinations of features, so they are better at fitting the pattern in the tail flow, but the predictions on the records from the head flow might be influenced by the noise. As a result, the ensemble of FDENs and GBDTs can take good care of both the head flow and the tail flow, thus making a significantly more accurate recommendation[2].

3.2 Feature Engineering

Here we describe some important and effective features we use.

3.2.1 Conditional Probability / Expectation Features

Music recommender systems often evolve a lot of categorical features, like `user_id`, `song_id`, `language`, `city`, `artist_name` and so on. Given one feature, the conditional probability of another feature can give rich information about users, songs, and the context.

For example, $P(\texttt{source\_type}|\texttt{user\_id})$ can describe whether the user listened to the song through the entry point that he was used to or not. $P(\texttt{source\_type} = \texttt{album}|\texttt{user\_id})$ and $P(\texttt{source\_type} = \texttt{online-playlist}|\texttt{user\_id})$ can also describe the habit of the user, which contribute a lot to the user portraits.

Similarly, for numerical features, we can compute the conditional expectations as features, and what's more, the conditional standard deviations can also help. For example, $E(\texttt{song\_length}|\texttt{user\_id})$ and $\sigma(\texttt{song\_length}|\texttt{user\_id})$ can be used to describe the preference and habit of a user.

3.2.2 SVD Features of Co-occurrence Matrices

As mentioned in the Introduction, the data is missing-not-at-random. So the co-occurrence matrices contain rich information. We construct a user-song co-occurrence matrix and a user-artist co-occurrence matrix, and use the Singularly Valuable Decomposition (SVD) algorithm to find the most important components as features. Specifically, to suppress the large values in the user-artist matrix, a calibration is adopted with the following equation:

$$r_{\text{user,artist}} = \begin{cases} 1 + 0.3 * \log(cnt_{\text{user,artist}}), & cnt_{\text{user,artist}} > 0 \\ 0, & cnt_{\text{user,artist}} = 0 \end{cases},$$

where $cnt_{\text{user,artist}}$ indicates how many times the user first listened to songs of the certain artist.

3.2.3 Timestamp Related Features

Since the data is ordered chronologically, we can use the index as the timestamp. This feature can help the model find the pattern evolution during the long period of time within the training set. Furthermore, we can also count the

[2]Codes are released at https://github.com/lystdo/Codes-for-WSDM-CUP-Music-Rec-1st-place-solution

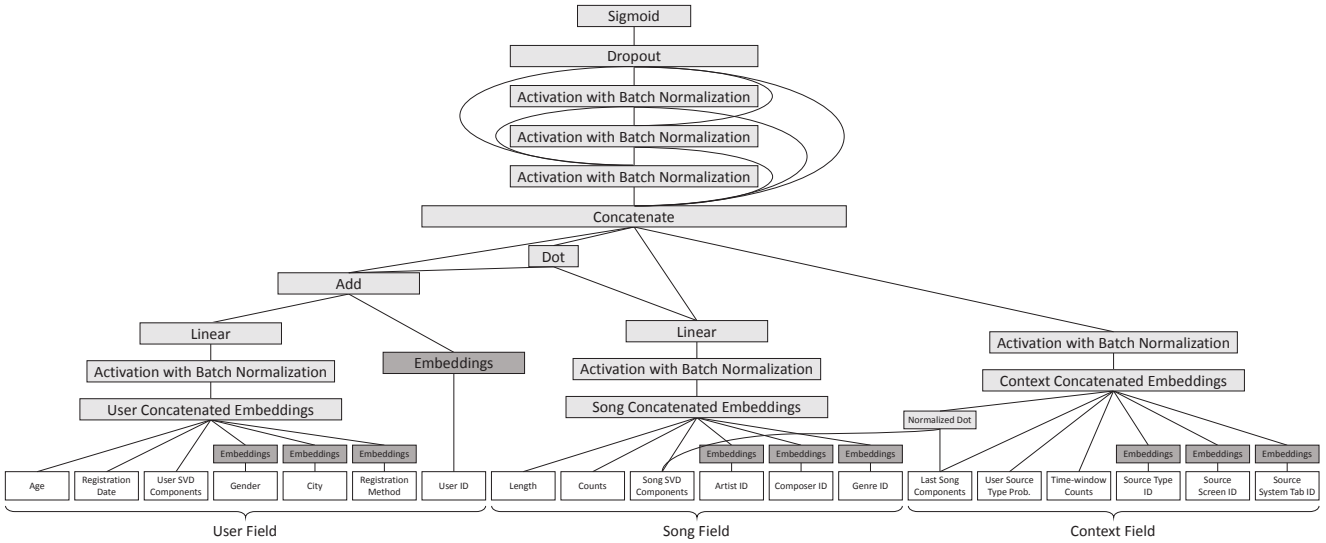

Figure 1: Overview of the proposed Field-aware Deep Embedding Networks. The inputs are divided into three fields, and the structures of different fields are slightly different. For demonstration, we only present a few features in the figure.

user/song activity within a time window regarding a record of the first observable listening event.

3.2.4 User Behavior Features

Since the algorithm is applied after the first listening event, we can obtain the songs that the user first listened to before and after the event was triggered. Using these features helps us describe the user's temporary interests, which are important for predicting the probability of recurring listening.

4. FIELD-AWARE DEEP EMBEDDING NETWORKS

In this section we, describe the structure of the proposed Field-aware Deep Embedding Networks, and the training method we used.

4.1 Overview of the Structure

The overview of the proposed Field-aware Deep Embedding Networks is illustrated in Figure 1. Minor details are omitted.

In the figure, "Activation with Batch Normalization" [8] stands for layers that perform the following transformation

$$\mathrm{ABN}(\boldsymbol{x}) = \mathrm{Activation}(\mathrm{BatchNormalization}(\boldsymbol{x}\mathbf{W} + \boldsymbol{b}))$$

and we used a variety of activations in the task. While "Linear" stands for layers performing linear transformations only, i.e.,

$$\mathrm{Linear}(\boldsymbol{x}) = \boldsymbol{x}\mathbf{W} + \boldsymbol{b}.$$

The layer "Dot" compute the inner product of the two inputs, i.e.,

$$\mathrm{Dot}(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{x}\boldsymbol{y}^{\mathrm{T}},$$

and the layer "Normalized Dot" compute the cosine similarity of the two inputs, i.e.,

$$\mathrm{NormalizedDot}(\boldsymbol{x}, \boldsymbol{y}) = \frac{\boldsymbol{x}\boldsymbol{y}^{\mathrm{T}}}{\|\boldsymbol{x}\|\|\boldsymbol{y}\|}.$$

4.2 Key Properties of the Network Structure

Here we discuss some important properties about the structure.

4.2.1 Field-aware Connections

The inputs are divided into 3 groups, i.e., user field, song field, and context field. The structure of each field is slightly different from each other. High-level features are extracted before they are concatenated together. This method can reduce the number of free parameters and prevent some unexpected feature combinations.

We also need to note that, there are some connections among the very bottom of the fields. For example, we compute the normalized dot products (i.e., cosine similarities) of the components of the currently-listening song and the last song the user listened to. These connections are built based on our intuitions.

4.2.2 Trainable Embeddings for Categorical Features

We use trainable embeddings with $\ell 2$ regularizations for categorical features, which can map the categorical features into a multidimensional space. The embeddings are initialized randomly, and trained through back propagations. Compared with one-hot encoding, this can also reduce the number of free parameters. With the help of shared weights in the latter layers and $\ell 2$ regularization, a better ability for generalization can be obtained.

4.2.3 Offsets for Users

In the user field, the final representations are not only based on the metadata of users. We allow an offset between the final representations and the predicted results from the metadata [1, 20].With the help of $\ell 2$ regularization, the model can automatically learn that, if the data for a user is sufficient, the representations of the user rely more on back propagations from the targets, while if not, rely more on the metadata.

We also tried allowing offsets for songs, but got little im-

provements, mainly because that the data for songs are far more sparse.

4.2.4 Dot Products for User-Song Pairs

Taking inner product can give better nonlinearity without adding too many free parameters. It can help fasten the convergence and give better results. Some work also suggests using element-wise multiplication or even outer product [13], but it requires more data to prevent overfitting.

4.2.5 Densely Connections at the Head Model

Inspired by the Densely Connected Convolutional Networks [7], we use densely connections after the layer where the outputs of the three fields are concatenated. Densely connections introduce direct connections between any two layers at the head model, thus can encourage more feature reuse and improve the parameter efficiency.

4.3 Model Training

In this model, we use the RMSProp algorithm with $\rho = 0.9$ as the optimizer. The learning rate decays at a fixed factor every epoch, i.e.,

$$lrate = lrate_0 * decay^{epoch\_num}$$

To make good use of the parallel computing power of GPUs and get predictions as fast as possible, the batch size is set to 8192. Apart from the batch normalization and the $\ell 2$ regularization for embeddings, we apply dropout [17] for regularization before the output layer. The dropout rate is set to 0.5.

Since neural networks are sensitive to the initialization, we use a bagging ensemble to get stable and repeatable results for validation. The best parameters are chosen according to local validation by random searching.

To encourage more diversity for the ensemble, we tried multiple forms of activations, including ReLU, LeakyReLU, PReLU, tanh and ELU. The numbers of hidden units also change accordingly.

5. GRADIENT BOOSTING DECISION TREES

Gradient Boosting Decision Tree is widely considered as one of the most powerful and commonly used machine learning techniques, and there are quite a few effective implementations, including XGBoost [2], pGBRT [19] and so on. In this paper, we use LightGBM [10], a highly efficient gradient boosting decision tree implementation by Microsoft Research. LightGBM accelerates the training process through Gradient-based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB), and can achieve almost the same accuracy with far less time consuming.

Unlike neural networks, GBDT splits the data into two parts in a tree node according to only one feature, thus some linear combinations of features can also help. We conduct some linear combinations of the features based on our intuitions. Except that, the features for FDENs and GBDTs are almost the same.

To encourage more diversity for the ensemble, we tried using weighted average of the predictions of GBDTs with slightly different subsets of features. The weights are tuned based on the Public Leaderboard. This method gives some significant improvements.

Table 1: Information about the Dataset

Item	Number
Records in the training set	7,377,418
Records in the testing set	2,556,790
Distinct users in total	34,403
Distinct users in the training set	30,755
Distinct songs in total	419,839
Distinct songs in the training set	359,966
Distinct artist in total	46,373
Distinct artist in the training set	40,583

6. EXPERIMENTS

In this section, we first introduce the dataset, then the validation setup is discussed. Procedures for feature selection and model tuning are presented later. The evaluation results and a visualization of artist embeddings are listed lastly.

6.1 Dataset

KKBOX is the leading music streaming services in Asia. In this challenge, we conduct experiments using the donated dataset from KKBOX. Detailed information about the dataset is summarized in Table 1.

Based on the observation from competitors, the dataset covers quite a long range of time, i.e., the training set is from Aug. 2016 to Jan. 2017, and the testing set is from Jan. 2017 to Feb. 2017. During the time, many new users joined the ecosystem, and many new songs emerged. As a result, in the testing set, 7.20% records involve users that do not appear in the training set, 12.52% records involve songs that do not appear in the training set, and up to 18.90% records involve users or songs that do not appear in the training set. This phenomenon calls for effective feature engineering from the metadata and user-song co-occurrence.

6.2 Validation Setup

As mentioned before, the training set and testing set are split based on time, and cover a long range of time. Therefore, we cannot assume that the pattern stays the same from the beginning to the end. As a result, cross-validation is not suitable for this problem.

Since the data is ordered chronologically, we use the last 20% data for validation, and when we generate the predictions for the testing set, all the data with labels are used to train the models.

To mitigate information leakage from the future to the past, when the features for validation are generated, only the records in the training set is used. This can help us get more reliable validation results.

6.3 Feature Selection and Model Tuning

For feature selection, we use the feature importance reported by LightGBM. Unimportant features are dropped directly. The best threshold is based on local validation. Our final models for LightGBM use about 400 features, most of which are SVD components, and conditional probability features. We use the first 48 components from user-song co-occurrence matrix, and the first 16 components form user-

Table 2: Evaluation Results on the Leaderboard

Method	Private AUC_{ROC}	Gain over Single Model	Public AUC_{ROC}	Gain over Single Model
Logistic regression with little feature engineering	0.66735	–	0.66527	–
Single FDEN	0.72787	–	0.72846	–
5-ensemble of FDENs with the same hyperparameters	0.73341	0.76%	0.73769	1.27%
5-ensemble of FDENs with different hyperparameters	0.73716	1.28%	0.73953	1.52%
25-ensemble of FDENs with different hyperparameters	0.73939	1.58%	0.74185	1.84%
Single GBDT	0.74277	–	0.74431	–
3-ensemble of GBDTs with slightly different feature sets	0.74389	0.15%	0.74569	0.19%
Ensemble of single FDEN and single GBDT	0.74390	–	0.74549	–
Ensemble of 25-ensemble FDEN and 3-ensemble GBDT	0.74695	0.41%	0.74916	0.49%
Our best submission before the deadline	0.74787	0.53%	0.75001	0.61%

artist co-occurrence matrix.

The best hyperparameters are obtained by random searching on the validation set, and at the validation stage the learning rate for GBDTs is 0.5[3]. When the predictions on testing set are generated, for LightGBM models, it takes about 8 to 10 hours for training with two Intel Xeon E5-2650 v2 with a learning rate of 0.1, and about 40 minutes for prediction. For a single FDEN model, it takes about 8 to 20 minutes with an NVIDIA GeForce GTX 1080, depending on how large the model is. Prediction generation with FDEN takes less than 1 minute.

6.4 Evaluation Results

Based on the aforementioned setup, we conduct experiments on the testing set. Results on the Leaderboard are listed in Table 2. For single FDEN models and 5-ensemble FDENs, we report the results with the best public score.

Before the results of the proposed methods are analyzed, we introduce the results of a simple logistic regression [9] as a baseline. This method uses the one-hot encoding results of 11 categorical features, including `user_id`, `song_id`, `source_system_tab`, `source_screen_name`, `source_type`, `city`, `gender`, `registered_via`, `artist_name`, `language` and `genre_id`, as well as two numerical features, including `age` and `song_length`. The hyperparatmer for $\ell 2$ regularization is tuned based on local validation. This simple method gives 0.66735 on Private Leaderboard, and 0.66527 on Public Leaderboard, which can help us gain a better knowledge of the dataset.

From the table, we can get that a single FDEN gives relatively poor results, while 5-ensembles can already give a very significant boost. Besides, we find that changing the hyperparamters, including the activation, number of nodes, $\ell 2$ regularization, learning rate and so on, can encourage more diversity between the predictions of FDENs, thus we witness the results of 5-ensembleof FDENs with different hyperparameters outperforms the 5-ensemble of FDENs with the same hyperparameters significantly, although the single models might be a little weaker. The 25-ensemble of FDENs is the 5-ensemble of 5-ensembles of FDENs with different hyperparameters, which means we repeat the experiments for 5 times with different random seeds, and then average the results. This gives about 0.30% more improvement on Private Leaderboard. Compared with the single model, 25-ensemble can improve the score by 1.58% on Private Leaderboard. We can get higher scores with even larger ensembles.

We need to note that, the hyperparameters are not optimized for the single model performance, so the results of single FDEN could be improved by using a larger model with stronger regularizations, smaller batch size and learning rate, and more epochs for training.

For GBDT models, we find that the results are a lot more stable and insensitive to hyperparameters compared with neural networks. To encourage more diversity, we tried training the models with slightly different feature sets. The differences between the subsets are mainly some count features with high importance based on the report by LightGBM. According to our experiments, 3-ensemble of GBDTs with slightly different feature sets can give about 0.15% improvement over a single GBDT.

During the evaluation, we find that the predictions of FDENs and GBDTs are quite uncorrelated. For example, the correlation coefficient between the predictions of 25-ensemble of FDENs and the predictions of 3-ensemble of GBDT is only 0.9156. Considering that the AUC scores are beyond 0.74, this is a very small correlation coefficient. We use Equation (1) to ensemble the results, and this gives an improvement of 0.41% over 3-ensemble of GBDTs, and an improvement of 1.02% over 25-ensemble of FDENs.

Our best submission before the deadline achieved 0.75001 on the Public Leaderboard, and 0.74787 on the Private Leaderboard, which finished the first place in the competition. The best submission is an ensemble of more models over different feature sets.

6.5 Embedding Visualization

In order to gain an intuitive understanding of the embeddings, we visualize the learned embeddings of the Top 25 most popular artist with t-SNE [12] in Figure 2[4].

As illustrated in the figure, *Fish Leong* and *Hebe* are very close to each other since they are both famous for sweet love songs, and *Eason Chan* and *Jacky Cheung* become close in

[3]Hyperparameters are also released with the codes.

[4]We drop the SVD components of user-artist co-occurrence matrix when training the models for visualization.

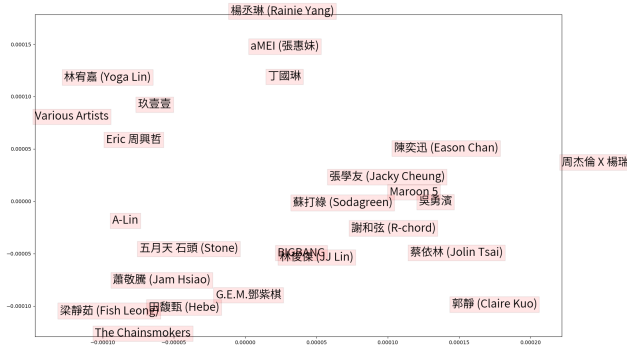

Figure 2: Visualization of the learned embeddings of the Top 25 most popular artist with t-SNE.

the middle because they both have magnetic male voices. Besides, *Stone* and *Jam Hsiao* are both famous for rock music, *Fish Leong* and *Stone* cover many songs with each other, *Rainie Yang* and *aMEI* are both singers with explosive expressions, and *Yoga Lin* and *Eric* are both young lyric singers. This shows that the learned embeddings can reflect the styles of artists to some extent.

7. CONCLUSION

In this paper, we describe our approach to the WSDM - KKbox's Music Recommendation Challenge. We use an ensemble of Field-aware Deep Embedding Networks and Gradient Boosting Decision Trees, with some feature engineering. Thanks to the nature of the two methods, we gain a very significant improvement after ensemble.

As the main goal of the challenge is to produce the best performance with the provided data, we did not consider the computation cost for generating the predictions. So the final predictions were based on a great many base models. Besides, since the data is time-sensitive, we found that supervised ensemble methods, like stacking ensemble and blending ensemble, didn't give better results. So the final framework for the ensemble is relatively simple. With the help of FDEN and GBDT, we were able to achieve an AUC score of 0.74787 on the Private Leaderboard finally.

Due to time constraints, we didn't try a lot of cross product features and methods of optimizing AUC directly [6], so there should be some room for further improvement under this framework. Another promising avenue of research is transfer learning. Since the data is missing-not-at-random, building neural networks to predict the first listening event and recurring listening event collectively might give some further improvement.

8. ACKNOWLEDGMENTS

This research has been partially supported by the National Nature Science Foundation of China (No.61673230).

9. REFERENCES

[1] B. Bai, Y. Fan, W. Tan, and J. Zhang. Dltsr: A deep learning framework for recommendation of long-tail web services. *IEEE Transactions on Services Computing*, 2017.

[2] T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794, 2016.

[3] H. T. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson, G. Corrado, W. Chai, and M. Ispir. Wide & deep learning for recommender systems. In *Proceedings of the Workshop on Deep Learning for Recommender Systems*, pages 7–10, 2016.

[4] Z. Cheng and J. Shen. On effective location-aware music recommendation. *Acm Transactions on Information Systems*, 34(2):1–32, 2016.

[5] M. Graczyk, T. Lasota, B. TrawiÅĎski, and K. TrawiÅĎski. Comparison of bagging, boosting and stacking ensembles applied to real estate appraisal. In *Proceedings of Asian Conference on Intelligent Information and Database Systems*, pages 340–350, 2010.

[6] A. Herschtal and B. Raskutti. Optimising area under the roc curve using gradient descent. In *Proceedings of International Conference on Machine Learning*, page 49, 2004.

[7] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[8] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of International Conference on Machine Learning*, pages 448–456, 2015.

[9] D. H. W. Jr and S. Lemeshow. Applied logistic regression. *Technometrics*, 34(1):358–359, 2000.

[10] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu. Lightgbm: A highly efficient gradient boosting decision tree. In *Proceedings of the Advances in Neural Information Processing Systems*, pages 3149–3157, 2017.

[11] E. Liebman, M. Saar-Tsechansky, and P. Stone. Dj-mc: A reinforcement-learning agent for music playlist recommendation. In *Proceedings of International Conference on Autonomous Agents and Multiagent Systems*, pages 591–599, 2015.

[12] L. V. D. Maaten and G. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(2605):2579–2605, 2008.

[13] Y. Qu, H. Cai, K. Ren, W. Zhang, Y. Yu, Y. Wen, and J. Wang. Product-based neural networks for user response prediction. In *Proceedings of the IEEE International Conference on Data Mining*, pages 1149–1154, 2016.

[14] R. L. Rosa, D. Z. Rodriguez, and G. Bressan. Music recommendation system based on user's sentiments extracted from social networks. *IEEE Transactions on Consumer Electronics*, 61(3):359–367, 2015.

[15] M. Schedl. The lfm-1b dataset for music retrieval and recommendation. In *Proceedings of ACM International Conference on Multimedia Retrieval*, pages 103–110, 2016.

[16] Y. Song, S. Dixon, and M. Pearce. A survey of music

recommendation systems and future perspectives. In *The International Symposium on Computer Music Modeling and Retrieval*, 2012.

[17] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[18] H. Steck. Training and testing of recommender systems on data missing not at random. In *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, Dc, Usa, July*, pages 713–722, 2010.

[19] S. Tyree, K. Q. Weinberger, K. Agrawal, and J. Paykin. Parallel boosted regression trees for web search ranking. In *Proceedings of International Conference on World Wide Web*, pages 387–396, 2011.

[20] H. Wang, N. Wang, and D. Y. Yeung. Collaborative deep learning for recommender systems. In *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1235–1244, 2015.

[21] Y. Wang, M. Bellus, J. F. Geleyn, X. Ma, W. Tian, and F. Weidle. A new method for generating initial condition perturbations in a regional ensemble prediction system: Blending. *Monthly Weather Review*, 142(5):2043–2059, 2014.

[22] G. Zhou, C. Song, X. Zhu, X. Ma, Y. Yan, X. Dai, H. Zhu, J. Jin, H. Li, and K. Gai. Deep interest network for click-through rate prediction. *arXiv preprint arXiv:1706.06978*, 2017.