Wide & Deep Learning For Music Recommendation

Kaho Chan
kc3137
kc3137@columbia.edu

Yu Wang
yw3025
yw3025@columbia.edu

Jingxi Xu
jx2324
jx2324@columbia.edu

1. Introduction

With the growth of digital music industry and popularity of music software and platform like Spotify, Google Play, etc., automatic music recommendation has become a hot research topic in the field of machine learning/information retrivial.

Two broad directions in music recommendation are *collaborative filtering* vs. *content-based recommendation*. While the collaborative filtering relies on the usage patterns (items that a specific user has consumed or rated which provide information about the user's preference more directly), content-based recommendation uses item content (audio signal) and metadata (artist, year of release, etc.). According to [Sla11], collaborative filtering generally outperforms content-based methods but it cannot handle cold-start problem (no usage data for new songs).

One challenge in developing recommendation systems is to achieve both *memorization* (learning the frequent co-occurrence of features and exploiting the available correlation in the historical data) and *generalization* (based on transitivity of correlation and exploring new feature combinations) [CKH+16]. Memorization can be achieved effectively by cross-product transformations while embedding-based models, such as factorization machines are useful methods to achieve generalization.

2. Related Work

Previous pioneering work presented by [HKV08] uses collaborative filtering with implicit datasets (no direct rating/preference information from users but only purchase histories) for TV shows recommendation. They transform the implicit feedback into preference-confidence paradigm and provide a latent vector algorithm to handle it. Inspired by their work, [VdODS13] adopts their *weighted matrix factorization* (WMF) and *alternating least squares* (ALS) optimization methods for content-based music recommendation. However, different from other content-based methods such as Mcfee et al [MBL12], who use metric learning to learn a similarity metric for comparing bag-of-words representations of audio signals, [VdODS13] adopts a deep convolutional neural network for predicting. They achieve a better performance and also handles the cold start problem so that new and unpopular songs can be recommended. Later on, Wang et al [WW14] develops a novel model which can unify the extraction and prediction into an automated simultaneous process, which further improves the performance of hybrid and content-based music recommendation systems.

Cheng et al [CKH+16] recently has presented a wide & deep learning framework for jointly training neural networks with embeddings and linear model with feature transformations, which effectively handles the challenge of both memorization and generalization. Their methods are the main baseline model we refer to in this project.

3. Problem Formulation

The dataset we use is provided by KKBOX. It is also used by a challenge from Kaggle where we apply our model in practice. For more information about the dataset and Kaggle challenge, please see here.

In this challenge, we train a model to predict the chances of a user listening to a song repetitively after the first observable listening event within a time window was triggered. The training set provides user-song pairs with the `target` variable indicating whether there is a repetitive listening activity of that user to that song within a month. The test set is some new user-song pairs not recorded in the training set where we need to predict their `target` variables. Some extra features like entry point, artist, song length are also provided in training set and extra sets.

Inspired by previous work presented in section 2, in this project, we aim to adopt the hybrid wide & deep learning framework as the baseline model. Then we will try different combinations and preprocessing of the features to implement both collaborative filtering and content-based methods to find an optimal configuration with best performance for music recommendation.

3.1. Criteria

The criteria used both in this paper and in the Kaggle contest is area under receiver operating characteristic curve (short for ROC-AUC or simply AUC).

AUC is the area under the receiver operating characteristic curve (ROC). The latter one works by plotting the true positive rate (TPR = TP/ (TP+FN)) against the false positive rate (FPR = FP / (FP + TN)) at various threshold settings.

4. Method

In this project we will use a Wide and Deep Network. A Wide and Deep Network combines a linear model with a feed forward neural network. It jointly training feed-forward neural networks with embedding layers and a linear model with both memorization and generalization.

In this section, we will talked about the model we are going to use as shown in Figure 1. We will omit some of the implementation detail here, which will be discussed in the next section.

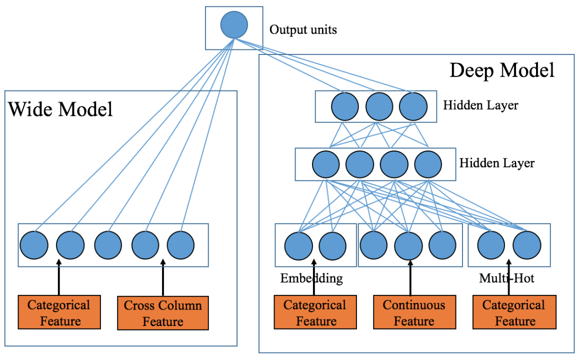

Figure 1. Wide and Deep model structure for music recommendation

4.1. Wide Model

The Wide Model is a linear model such as a linear regression. It follows $y = w^T x + b$ where x is the feature vector. The input feature in the wide model part contains the sparse feature generated from sparse base columns and crossed column features.

Sparse columns include categorical features such as the language of the song. We transform the categorical features with fewer categories directly using feature hashing to make it an vector with fixed length. For other categorical columns which have many more options than we want to put keys, we hash them into buckets to limit the categories. Note that although there might be many collisions during hashing, this technique works well in practice because the model still have other features to separate the content of each group.

The crossed column feature is generated by hashing the combination of categorical columns into real value variables. For example, after crossing the language of the song and city of the user, we will have a mapping as:

```
            ...
(City 10, Language 15) -> 42
(City 13, Language 20) -> 97
            ...
```

Wide models with crossed feature columns can memorize sparse interactions between features effectively. However, one limitation is that it cannot generalize feature combinations that never appear in the training data. Thus we introduce the deep model.

4.2. Deep Model

The Deep Model is a fully-connected feed forward network with two layers of hidden units. It contains the continuous feature, densely-embedded vector from each of the sparse, high-dimensional categorical features and multi-hot transformation of categorical feature with low dimensions. These features are combined together as input layer to the hidden units in the forward pass and use rectified linear units (ReLUs) as activation function for each layer.

$$a^{l+1} = f(w^l a^l + b^l)$$

Through dense embedding, deep models can generalize better and make predictions on feature pairs that were previously unseen in the training data.

4.3. Joint Training

The wide models and deep models are combined by weighted sum of their final output log odds, then feeding the prediction to a logistic loss function. Then the model uses gradient descendent methods to update the parameters simultaneously for the wide and deep model. For detailed implementation of the model, see part 5.3.

5. Preliminary Experiments

5.1. Code Repository

Our code is shared through private Github repository.

5.2. Features

As we are training on a real-world data, features will take a very important part in the performance.

5.2.1 Raw data

The features given in the data set is listed as following:

- **Meta data** The data about the event, including user id, song id, source screen name, source type and source system tab. The last three are information about where does the event happen (where does the user play the music for the first time.)

- **Song related data** Associated with the song id, including: the length of the song, the genre categories of the song (may have multiple genres), artist name, composer, lyricist, language of the song, and year of the song.

- **Member related features** City of the user, age of the user, gender of the user, where does the user registered, the registration time and expiration date.

5.2.2 Feature Cleaning

The data is not very complete and we have to fill in the empty slots and clear the noise in the feature table. To be specific, we now use an tag 'Unknown' for each categorical feature, except that for missing song length, we fill in with the mean time and for missing language of song, we fill in with the mode.

5.2.3 Feature Engineering

As a first attempt, we do the following feature engineering.

- We calculate the validate days using expiration date and registration date and also break the two dates into three parts respectively: (year, month, day).

- We convert some of the numerical data (song length and user age) into categorical data by 'bucketization', that is, given a set of boundaries $b_0 = -\infty, b_1, \cdots, b_k, b_{k+1} = \infty$, split the data into categories represented by different bins $(b_i, b_{i+1}), i = 0, 1, \cdots, k$.

- We create new features based on raw data, including 'genre count' for the number of genre ids of one song, 'lyricist count' for the number of lyricists.

- To pass in the deep learning part for tensorflow, we have to make the sparse categorical data dense, so we pass the features through embedding layers.

5.3. Environment

The experiment of logistic regression is implemented by using 'sklearn.linear_model.LogisticRegression' model in scikit-learn package. And the wide and deep network is implemented by using tensorflow estimator 'tf.contrib.learn.DNNLinearCombinedClassifier'.

5.4. Configuration

For the wide part, we include all the categorical features mentioned above, together with crossed features, including:

- User id and song id

- Registration year and expiration year

- User id and language of the song

- User age (bucketized) with artist name

Then we concatenate all the features as one vector and pass it into the wide part.

For the deep part, we use all the categorical features by embedding them into dimension of 8, together with numerical features (age, song length, genre count and lyricist count). For user id, song id and artist name, we also include 64-dimension embedding features respectively. Then we concatenate all the features as one vector and pass it into the deep part.

We use two layers of densely connected hidden layers for the deep part. The dimensions of the two layers are 100 and 32 respectively.

To train the model, we use default optimizer as provided by given API, i.e. Adagrad optimizer for the deep part and FTRL optimizer for the wide part.

5.5. Results

Table 1 presents the preliminary results of three models using logistic regression, gradient boosting and wide & deep model respectively. The test AUC is validated by the public leader board on Kaggle.

Model	Train AUC	Test AUC
Logistic regression	0.625	0.543
Gradient boosting	0.750	0.655
Wide and deep	0.782	0.609

Table 1. Preliminary Results

5.6. Discussion

We can see from table Table 1 that the simple model using only logistic regression has the worst performance in both training and testing stages. Also, while using gradient boosting has the best test result, its training accuracy is slightly worse than our wide & deep model. These results are understandable in the context of our preliminary experiments.

Gradient boosting model is achieved by taking advantages of the existing Python `lgbm` package, it has already been well developed and ready-to-use. There are lots of implicit optimization going on within these functions we called. Even though the wide & deep model should theoretically generate a better result than pure linear models as it tries to achieve both memorization and generalization, the feature engineering and data cleaning/pre-processing for wide & deep are not mature enough in the preliminary settings so that it does not outperform gradient boosting model. As for now, our wide & deep model is more of a content-based deep neural network prototype than a complete one as in [CKH+16], and more experiments need to be done to find a better configuration of the deep neural network structure.

Logistic regression model is developed from scratch by us. It does not have as advanced a structure as wide & deep model, neither has it been well developed for industry use; therefore, it has the worst result of all three.

6. Next Step

For preliminary results, we have not yet done thorough experiments on both features and structures. In the following weeks, we will try to clean the feature better and experiment with different structures to achieve a better result.

References

[CKH+16] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. Wide & deep learning for recommender systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*, pages 7–10. ACM, 2016.

[HKV08] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, pages 263–272. Ieee, 2008.

[MBL12] Brian McFee, Luke Barrington, and Gert Lanckriet. Learning content similarity for music recommendation. *IEEE transactions on audio, speech, and language processing*, 20(8):2207–2218, 2012.

[Sla11] Malcolm Slaney. Web-scale multimedia analysis: Does content matter? *IEEE MultiMedia*, 18(2):12–15, 2011.

[VdODS13] Aaron Van den Oord, Sander Dieleman, and Benjamin Schrauwen. Deep content-based music recommendation. In *Advances in neural information processing systems*, pages 2643–2651, 2013.

[WW14] Xinxi Wang and Ye Wang. Improving content-based and hybrid music recommendation using deep learning. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 627–636. ACM, 2014.