Practical 3: Preferences in Streaming Music

Yi Ding, Linying Zhang, Danny Zhuang
Kaggle: LYD

April 7, 2017

1 Technical Approach

- **Feature Engineering**
  - **Artist features** Using Musicbrainz library, we were able to find the Wikipedia link of the 2000 artists, and extracted genres of each artist from Wikipedia. There are 662 genres in total, and each artist is associated with 3.4 genres in average. Noticing that 586 genres have fewer than 20 artists, we dropped these genres from our feature space, and for those artists who now have no association with any genre, we assigned them to a group called minority.
    Using the training set, we also extracted features related to the popularity of artists: (1) number of followers: the total number of users listening to each artist based on the training data; (2) plays per artist: the total number of plays by all users in the training data.
    After feature engineering for artists, we have a 2000 x 665 matrix for artists, including artist ID, genre and popularity for all 2000 artists.

  - **User features** We first imputed the missing data in the user profiles: creating variable unknown gender for users missing sex, and using the median age to impute users with missing age. We then one-hot encoded categorical features, including sex and country. We then characterized user activity by: (1) summing the total number of artists each user listens to based on the training data and (2) summing the total number of plays each user has across artists in the training data. By then, we had a 233286 x 246 matrix, including user ID, age, gender, country and user activity for all users.

- **Model Building Proccess**
  - **Variations of user-median model**
    We made some variations on the user-median model by including artist-median. First we tried to predict user not in the training set but their paired artist in the training set by artist median instead of global median. After we split train:validation = 7:3, we found that only a few such user-artist pairs existed, indicating that this variation of the user-median model would almost be identical to the user-median model.

Then we tried a second variation by predicting user-artist pairs with both user and artist present in the training data by weighting user-median and artist-median, with (user, artist) weights (0.8, 0.2), (0.7, 0.3), (0.6, 0.4), and (0.5, 0.5).

– **Cluster-cluster median model**
We tried using user-cluster median and artist-cluster median instead of individual artist and user median for prediction. The idea behind using cluster rather than individual is that predicting plays by individual median may overfit the data. We first cluster artists based on gernes using Kmeans, and tuned K from 100 to 700. To better cluster users, we leveraged the artists cluster we obtained. For each user, we created another 200 features each of which is an artist cluster. We then summed the number of plays of all artists in a given cluster for each user based on training data. We added these 200 features to the original user dataframe and did Kmeans clustering on users. We tried K=2, 10, 50, 100, 500, and looked at the histogram of user distribution to pick the number of clusters. We then calculate the median for each user cluster and artist cluster pair, which end up to be 20,000 pairs. The prediction is done by finding the the cluster of both user and artist, and using the cluster-cluster median as our prediction.

– **Topic Modeling**
We suppose theres a latent type for users, with distribution of each type as $\boldsymbol{\theta}$, and each type of users corresponds to a specific probability of listening to a specific artist as $\boldsymbol{\beta}_k$. Thus we adopted topic modeling and EM algorithm to estimate $\boldsymbol{\theta}$ and $\boldsymbol{\beta}_k$. To reduce computation time, we clustered users into 100 groups and use Nave Bayes to estimate $\boldsymbol{\theta}_{init}$ and $\boldsymbol{\beta}_{init}$ and fed them into topic model as initial value of $\boldsymbol{\theta}$ and $\boldsymbol{\beta}_k$. Then in each E step, we estimated the probability of each user belongs to each state based on the number of plays he listened for each artist. Then in the M step, we updated parameters $\boldsymbol{\theta}$, $\boldsymbol{\beta}$ to maximize the likelihood of observing the data, where $\boldsymbol{\theta}$ is a 100*1 vector encoding the distribution of clusters, and $\boldsymbol{\beta}$ is a 100 * 2000 vector with each row encoding the probability of artist being listened given user cluster. Given user i and artist j pairs, we predict based on three steps: First, compute probability of users being to each group and assign it the largest probability cluster k; second, estimate total plays of that user $\hat{n} = \frac{\text{total non-zero plays}}{\text{total probability corresponds to non-zero plays}}$. Finally, predict plays for user i and artist j pair as $\hat{n} \times \beta_k j$.

– **Weighted-mean-deviation**
We also did weighted-mean-deviation modeling to predict number of plays. Basically, we clustered users into different groups as described above. Then we looked at only the users in the same user cluster who have listened to the given artist . We then take a weighted-mean-deviation of each of the similar users' plays of the artist, weighted by the similarity between the user we are predicting on and the users in the training data.

– **Matrix factorization and Completion** We reformulate this predication problem as imputing missing data for a 233286 x 2000 matrix. Given the sparsity and low rank nature of this matrix, we applied generalized low rank model proposed by Udell Madeleine,

et al to conduct matrix factorization and completion.

– **Generalized linear mixed model** Regression method is considered because plays can be treated as continuous outcome. However, linear regression could not be used in this case because the instances are highly related in our data. Inspired by Genome-Wide Association Study (GWAS), which includes population structure as random effects to account for individual relatedness, we included user and artist as random effect to account for instance relatedness and also include sex, country and age as fixed effects according to the routine of GWAS.

$$\boldsymbol{Y} = b_0 + b_{sex}sex + b_{age}age + b_{country}country + b_{user}user + b_{artist}artist + \epsilon$$

$$Var(\boldsymbol{Y}) = \boldsymbol{U}_{user}Var(b_{user})\boldsymbol{U}_{user}^T + \boldsymbol{U}_{artist}Var(b_{artist})\boldsymbol{U}_{artist}^T + Var(\epsilon)$$

The term $\boldsymbol{U}_{user}Var(b_{user})\boldsymbol{U}_{user}^T + \boldsymbol{U}_{artist}Var(b_{artist})\boldsymbol{U}_{artist}^T$ are the covariation matrix which accounts for instance relatedness.

2 Results

Model	MAE on validation set	MAE on test set
MODEL BEATS BASELINE		
Generalized linear mixed model	-	135.818
MODELS FAILED TO BEAT BASELINE		
Weighted user-artist-median model	140.77	139.38
Weighted-mean-deviation	107.12	159.03
MODELS FAILED TO IMPLEMENT		
Topic modeling		
Matrix factorization and completion		

Table 1: Mean absolute error of models tested on either validation set or on kaggle test set.

We tuned Kmeans artist clustering with K=100, 300, 500 and 700, and based on the Inertia vs K graph, we estimated the "elbow point" to be aroun K=200, where the marginal gain of decrease of intertia becomes substantially less.
For user clustering, we tried Kmeans with K=2, 5, 10 and 100. When K=10, almost all users get to group into one cluster, and to find a balance between efficiency and clustering accuracy, we chose K=100 where about half of the users belong to one cluster, and other half users were distributed in the other clusters.

For the weighted user-artist median model, we splited train/validation on 0.7/0.3 ratio, where we tuned the weights for user-median and artist-median from (0.8, 0.2) to (0.5, 0.5) with 0.1 increment,

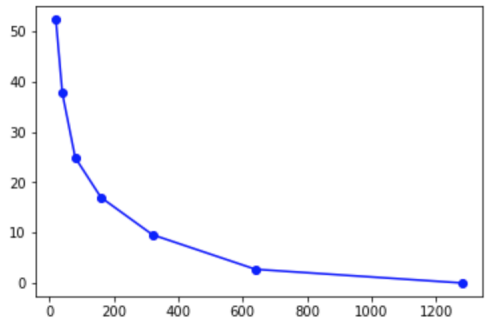

Figure 1: Tuning K for artist clustering in Kmeans.

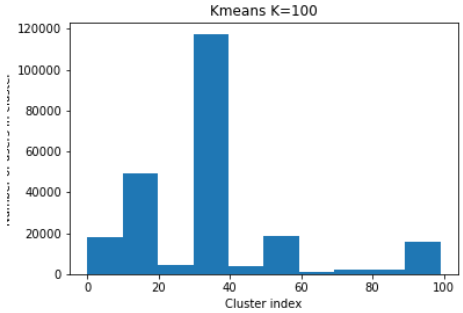

Figure 2: User distribution across 100 cluster from Kmeans.

the MAE on validation set was reported in Table 2, from which we saw that adding the linear effect of artist worsen our prediction, which was contradictory to our expectation, indicating the effect of artist on plays may not be linear.

Weight: User, artist	MAE on validation set
0.8, 0.2	140.77
0.7, 0.3	143.54
0.6, 0.4	147.81
0.5, 0.5	153.56

Table 2: Weights tuning of weighted user-artist median model.

The weighted-mean-deviation method was done with 0.6 regularization and without regulartion, which obtained MAE on validation set 108.95 and 107.12 respectively. The best kaggle MAE using this approach was 159.03.

We spent a significant amount of time on topic modeling, and have the EM algorithm implemented and tested on simulated data. However, when we ran it using the training data, the probability of artist being listened in each user cluster become extremely small that numpy outputs NaN instead. We tried some numeric methods, such as log transformation, and rerun the model. However, the convergence was slow and each EM cycle took about 10 minutes so we were not able to get result for this model.

The matrix factorization was not completed due to extremely high memory requirement and long computation time.

From generalized linear mixed model, we obtained MAE of 135.818 on kaggle public leaderboard. Due to the limited time and high computational complexity, we did not get time to do cross-validation for this model.

3 Discussion

We were very attracted to the idea of applying topic modeling into music play prediction. When developing and testing topic modeling, we kept getting NaN error even when the logic of algorithm is right. One cause for NaN is that when x_{ij} is too large $q_{ik} = \theta_k \times \prod_{j=1}^{v} \beta_{kj}^{x_{ij}}$ will be stored as zero by Python. We solve this problem by using $\log(q_i k)$, and calculate normalize q_{ik} as $\log(\frac{q_{ik}}{\sum_{k=1}^{c} q_{ik}}) = \log q_{ik} - \log(\sum_{k=1}^{c} \exp \log q_{ik})$ and use $scipy.misc.logsumexp()$ function to maintain numerical stability. To keep everything as logs for as long as possible, we also calculate $\log \boldsymbol{\beta}$ and $\log \boldsymbol{\theta}$. But it incurs another NaN problem, when calculating β_{kj}, we need to calculate $\log(q_{ik} \times x_{ij}) = \log x_{ij} + \log q_{ik}$, NaN occurs when $x_{ij} = 0$, so we add 1 to each data point to solve this problem. After solving these two problems, the function worked well and converged quickly for simulated data. However when applying to our train data, each EM step took about 10 minutes to complete so we were not able to get results before the deadline.

Matrix factorization and completion is a popular method for user recommendation system, but the heavy computation requirement makes it impractical for our data given a limited time. We submitted our job to Odyssey and requested 40GB memory for 10 hours, but it exited out of time.

Through out the model building process, we tried various methods to get better clustering on users and artists, which we initially thought could reduce the computational complexity in downstream model fitting and prediction, and avoid the risk of overfitting. However, comparing all the models with cluster involved to a similar model using individual user and artist directly, we found that the model with individual subject was always better. Although we didn't fully rationalize why it is the case, but given the user-artist pair we had was only a small portion of all possible user-artist pairs, grouping them may result in significant loss of information.

4 Reference

[1] A Probabilistic Model for Music Recommendation Considering Audio Features. Springer Berlin Heidelberg. Li, et al. 2005
[2] Udell, Madeleine, et al. "Generalized low rank models." Foundations and Trends in Machine Learning 9.1 (2016): 1-118.
[3]Price, Alkes L., et al. "New approaches to population stratification in genome-wide association studies." Nature Reviews Genetics 11.7 (2010): 459-463.