EMI Music Data Science

Metehan Dizioglu, Rolando Vicaria

Introduction

The question this project attempts to answer is ‘can we predict if a listener will like a new song?’ This problem was presented on Kaggle by EMI, one of the world’s largest music production companies. We attempt to answer the question by looking at listener demographic data collected by EMI as well as answers to questionnaires about music preferences. This data was collected by EMI and its affiliates throughout the UK. It includes men and women ranging from 14 to 94 years of age. All records were anonymized; users, artists, and tracks were all given unique identifiers.

Data

The data set was initially split up across several CSV files. The first of these was ‘users.csv’ which contained the following information for ~49,000 individuals:

- User id - anonymized user identifier
- Gender - In text format
- Age - Integer, not age groups, there were missing data.
- Employment status - In text format, there were missing data.
- Geographical Region - Different regions in UK.
- Importance of music in user’s life - In text format, there were missing data.
- Hours spent in a day listening to music they own - In text format, there were missing data.
- Hours spent in a day listening to background music - In text format, there were missing data.
- 19 Questions asking user to rate on scale of 1-100 whether they agree with the statement
    1. I enjoy actively searching for and discovering music that I’ve never heard before

2. I find it easy to find new music
3. I am constantly interested in/and looking for more music
4. I would like to buy new music but I don’t know what to buy
5. I use to know where to find music
6. I’m not willing to pay for music
7. I enjoy music primarily from going out to dance
8. Music for me is all about nightlife and going out
9. I’m out of touch with new music
10. My music collection is a source of pride
11. Pop music is fun
12. Pop music helps me to escape
13. I want a multimedia experience at my fingertips wherever I go
14. I love technology
15. People often ask my advice on music/what to listen to
16. I would be willing to pay for the opportunity to buy new music pre-release
17. I find seeing a new artist/band on a tv a useful way of discovering new music
18. I like to be at the cutting edge of new music
19. I like to know about music before other people

The second CSV file, ‘words.csv’, contained information about what words users used to describe artists. It did this for over 100,000 combinations of users and artists:

- User id - Integer
- Artist id - Integer
- Heard of - In text format, there were missing data.
- Own artists music - In text format, there were missing data.

- Like artist – Integer between 1-100, there were missing data.
- 82 words that a user could choose from to describe the artist, e.g. Cheesy, Soulful, Aggressive, etc.

The third file, ‘train.csv’, contained several rows, each with a user id, artist id, track id, and a rating. The rating was on a scale of 1-100, where 100 indicated strongly liked the song. This is the target variable that we wish to predict, the rating a listener will give to a new track they have never heard before.

The last file, ‘test.data’, contained the same data as the ‘train.csv’ but rating missing, which is our target in this project.

Initial attempts

We began by branching out and trying two different approaches in parallel. One approach was a linear SVM and the other was k-means.

Our first attempt at running k-means was for the user’s gender, age, and region. We did this for 10 clusters. For each cluster, we aggregated the artists that the users in that cluster had stated that they like. Instead of predicting a rating from 1-100, we tried to predict a “yes” or “no” for each track based on whether or not the artist was contained in the list of artists for that user’s cluster. We measured accuracy by assuming that an actual rating > 50 was a “yes” and < 50 was a “no”. We then compared our predictions to the transformed actual ratings. We achieved 22% accuracy in this way.

We attributed this low percentage to the fact that the artist list for each cluster was pretty large. Therefore we had an overwhelming number of “yes” predictions.

We initially attempted to run an SVM using only the user’s gender, age, and region together with the artist id and track id.

We measured accuracy by counting the number of predicted ratings that were within 20, 10, and 5 percent of the actual ratings. This initial SVM gave the following results:

% difference of prediction from actual rating	% of test data correctly predicted
Within 5% of the actual rating	8%
Within 10% of the actual rating	32%
Within 20% of the actual rating	56%

Factorization Machines

We decided to use Factorization Machines[1]. Factorization Machines (FM) are a new model class that combines the advantages of polynomial regression models with factorization models. Inference in factorized polynomial regression models can be understood as an intertwined combination of a Principal Component Analysis like feature selection step and a linear regression step on the resulting latent feature space representations. Like polynomial regression models, FMs are a general model class working with any real valued feature vector as input for the prediction of real valued dependent variables as output. However, in contrast to polynomial regression models, FMs replace two-way and all other higher order interaction effects by their factorized analogues. The factorization of higher order interactions enables efficient parameter estimation.

We chose to use libFM[2] library for our project. libFM is a software implementation for factorization machines that features stochastic gradient descent (SGD) and alternating least squares (ALS) optimization as well as Bayesian inference using Markov Chain Monte Carlo (MCMC). In our analysis we applied all three of SGD, ALS, and MCMC algorithms as learning methods.

Each row contains a training case (x; y) for the real-valued feature vector x with the target y. The row states first the value y and then the non-zero values of x. The libFM tool trains a Factorization Machine (FM) model from training data -train and predicts the test data -test.

Processing Data

We had to process the data and handle missing fields and then shape it such that we can run our learning algorithms on it. We started from ‘users.csv’. This file contained user information and answers to questions directed by the surveyor as described above.

We calculated the mean age and populated the missing age information with the mean. Then we created age groups (20-25, 30-35,..etc). This gave us 20 new columns, we marked the column with 1 if the user was in that age group, 0 if not.

By looking at all the rows for ‘Work Status’ we found out all the unique work statuses and again created a column for each unique work status. Iterated through all the users (rows) and mark the column with 1 if that user was in that work status, 0 if not. We applied the same logic also to both ‘Region’ and ‘Music’, essentially we ended up with unique columns for each unique response that user gave.

This approach got a little harder when we start processing the hours the user spends listening to the music they own and the hours the user listens to background music. The answers were not clearly indicated and used words like ‘1’, ‘1 hour’, ‘1 hours’, ‘one hour’, ‘one’ which all meaning the same thing. This data had to be translated into integers. We used some Python and some R scripting to process these entries.

The next data file we processed was ‘words.csv’. We first calculated the means of each column and filled in the missing data in the columns with respective means. There is a column associated with each of 82 words but we applied the same techniques we used for ‘users.csv’ for columns ‘heard of’, ‘own artist’, which generated several columns since answers were like, ‘own none’, ‘own a little’, ‘own bunch’..etc.

At the final steps we aligned the both the ‘newusers.csv’ and ‘newwords.csv’ with ‘train.csv’ and ‘test.csv’ individually. The final format of the ‘train.new.csv’ is [user, artist, track, userdata, worddata]. Our features totaled 262 in our combined ‘train.csv’ and ‘test.csv’ files.

Finally, the data needed to be reformatted into a structure that could be fed into libFM. This format is the same one used by other machine learning libraries, namely, LIBSVM and SVMLite.

First we fitted several latent factor models to the target. For each artist, track and user, we have a separated column indicating whether this rating is from this user (of this track or of this artist). Thus, each row in the data matrix contains exactly 3 ‘1’s, representing artist, track and user respectively. The main idea of latent factor model is to capture the interaction between users and tracks. Tracks can be categorized into several groups and so are users. Users in the same group tend to have similar taste of different kinds of music. The latent factors are used to model users’ different preferences of music from different categories. We used libFM to train this part of model.

We used a cross-validation technique where we split the training data into 10 equal parts. The final prediction was the averaged result of the 10 cross-validation runs.

The error reported here is the result returned from submitting the predictions generated from the models on Kaggle.

Learning method	Root mean squared error (RMSE)	Iterations	Initial Std Dev
SGD	15.28368	1000	0.5
MCMC	15.20170	500	1
ALS	15.34847	500	200

Had these predictions been submitted while the competition was active, these would have qualified for 36th, 33rd, and 41st place respectively.

Gradient Boosting Machine

As a final optimization, we experimented with the gradient boosting machine[3] (gbm) package in R. We took the output of the factorization machine predictions and ran it through a linear regression. The output of that was then fed into the gbm library. There was immediate improvement on the RMSE results. At 20 iterations (n.trees on gbm), the RMSE of the SGD predictions improved to 14.93113 (a hypothetical 31st place had it been submitted while the competion was active). At 200 iterations, this improved to 14.05957 (hypothetical 16th place). At 1000 iterations, this improved to 13.52163 (hypothetical 10th place). We did not attempt more 1000 iterations because of the long running time of this algorithm.

References

1. http://www.ismll.uni-hildesheim.de/pub/pdfs/FreudenthalerRendle_FactorizedPolynomialRegression.pdf
2. http://www.libfm.org/
3. Friedman, J. H. : Stochastic Gradient Boosting, http://www-stat.stanford.edu/~jhf/ftp/stobst.pdf