Music Rating Prediction: Final

Brian Doolittle and Pratap Luitel

COSC 174

Problem:

Our goal is to predict how a user will rate a song given their previous ratings, user profile, and artist descriptions. This project was originally posted on kaggle.com as a 24 hour hackathon taking place on July 21st, 2012 [1]. The winners of the competition were able to achieve a root mean square error (RMSE) of 13.196 [2].

For our final algorithm we implemented a latent factor model using matrix factorization. We chose to use this model because of its success in state of the art recommender systems as well as in the Kaggle competition [2,3].

State of the Art:

Recommender systems are commonly used to predict how a user will rate a new item. They perform collaborative filtering to identify items that a user may rate highly. Collaborative filtering only requires users' past histories to make predictions. One technique used in collaborative filtering is matrix factorization. This latent feature model finds features for each user and item. To predict how a user will rate an item, one can take the inner product between the user and track latent features. These latent features can be learned by applying matrix factorization to the known ratings. One problem that all recommender systems face is the cold start problem. The problem is: given a new user or item, how can we make a prediction of its rating. Since no latent features have been learned for this user or item, they must be determined by comparison with other users in the training set [3].

Data:

Our data is a subset of EMI's 1 million interview dataset [4]. We have 3 data sets, "Users.csv", "Words.csv" and "Train.csv". Users.csv contains a user profile, Words.csv

contains user's description of artist, and Train.csv contains user ratings of tracks. Our data was taken over a 2 year period and contains 50,928 ratings of 184 tracks by 50 artists.

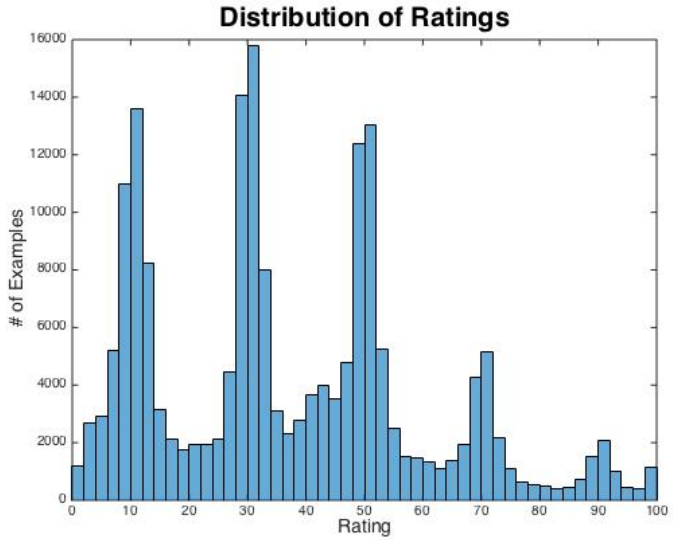

Fig. 1: The histogram shows the distribution of user ratings for the entire data set provided

The distribution of user ratings (fig. 1) shows an interesting trend. Rather than being centered around an average rating as one would expect, there are distinct peaks occurring at 10, 30, 50, 70 and 90 indicating that users prefer to rate these values. For a model to perform well on the data set, it would need to be able to predict these trends.

Preprocessing:

Words.csv and Users.csv contain numeric and string valued data. To incorporate these data sets into our model, we needed to convert the non-numerical fields into numerical values. We separated all strings into classes and represented each class with a binary feature. Fields with numeric ratings were normalized to 1. A tabled description of data manipulation can be found in appendix A.

In order to implement matrix factorization we needed to perform further preprocessing. We created matrix $M \in \Re^{[nUsers\ x\ nTracks]}$ where $M_{i,j}$ contains the user(i)'s rating of track(j). Since users only rate a handful of tracks, M is a sparse matrix meaning that it only has a few known entries.

To make our user profiles easy to access, we ordered them in matrix $UserProf \in \Re^{[nUsers\ x\ nFeatures]}$. However, some users are missing profiles. In these cases, we assigned the user the average for all user features. We applied similar preprocessing to the artist descriptions ordering them into matrix $ArtistProf \in \Re^{[nArtists\ x\ nFeatures]}$. Since multiple users rated each artist, the profile of each artist contains the average descriptions by all users.

Model:

Baseline Model: Average

For a baseline model we predicted the user's rating by assigning the training set average for the given track. In the case that there are no instances of the example track in our training set, our algorithm output the value of 30, the mode for track ratings. Testing this model in a cross validation we found an RMSE of 21.47.

Final Model: Matrix Factorization

Training the model

The goal of our latent feature model is to learn the matrix factors of M, $U \in \Re^{[nUsers\ x\ nLatent]}$ and $T \in \Re^{[nLatent\ x\ nTracks]}$. Rating predictions are determined by the inner product of the user latent features and the track latent features.

$$M_{i,j} = U_i \cdot T_j \qquad (1)$$

Our learning objective was to minimize the RMSE of our prediction.

$$E = \tfrac{1}{2}(M_{i,j} - U_i \cdot T_j)^2 + \tfrac{\lambda_1}{2}\|U_i\|^2 + \tfrac{\lambda_2}{2}\|T_j\|^2 \qquad (2)$$

We used regularization terms, λ_1 and λ_2, to help prevent overfitting and give us more control over our model. We performed an alternating least squares algorithm on our initialized latent feature with the following update parameters [3].

$$U_i \leftarrow U_i + \gamma(e_{i,j} * T_j - \lambda_1 U_i) \qquad (3)$$

$$T_j \leftarrow T_j + \gamma(e_{i,j} * U_i - \lambda_2 T_j) \qquad (4)$$

Here, $e_{i,j}$ denotes the difference of the actual and predicted ratings. On each iteration of our algorithm we update latent feature vector T_j, make a new prediction then update latent user vector U_i. All elements of U_i and T_j are updated in parallel to increase efficiency of our algorithm.

Since matrix factorization is a complex problem with many local optima, initial conditions to the learning algorithm are important. Through trial and error we found that initializing the user latent features with their 92 feature user profile. Tracks were initialized to ones.

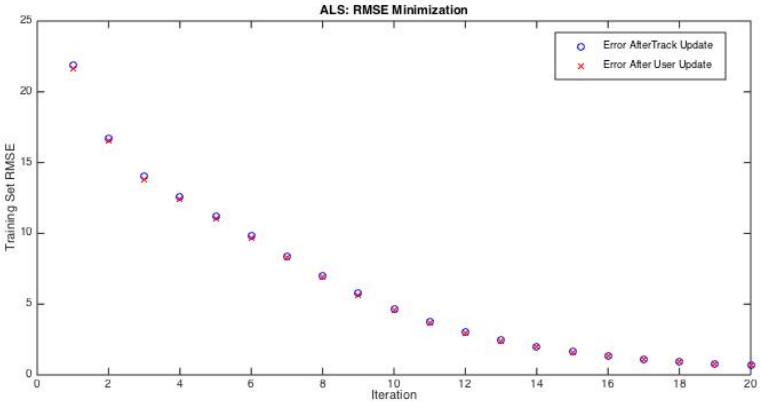

Fig 2: The plot shows the RMSE for each iteration of our ALS optimization. Blue shows the error after updating the track and red shows the error after updating the user.

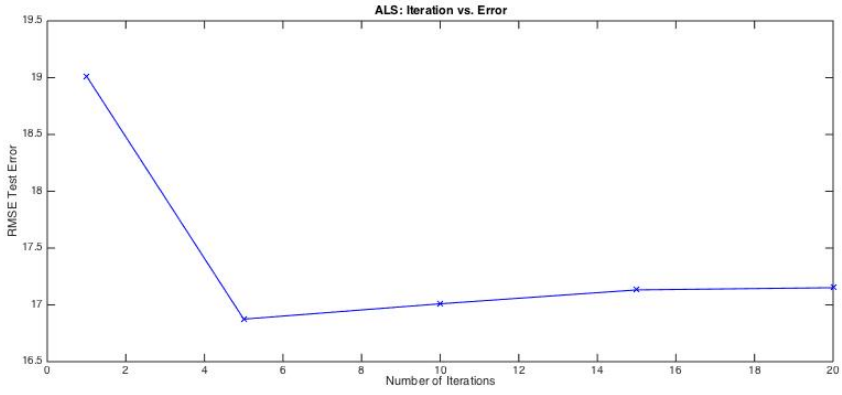

Fig 3: The plot shows the test error for different number of training iterations.

Fig. 2 shows that the algorithm reduces error on each step and thus, converges. Fig. 3 supplements this information showing the optimal amount of iterations to use is 5. Using less than 5 iterations results in high error because the model has not been sufficiently trained. Using more than 5 iterations actually results in more error because the algorithm begins to overfit the training data.

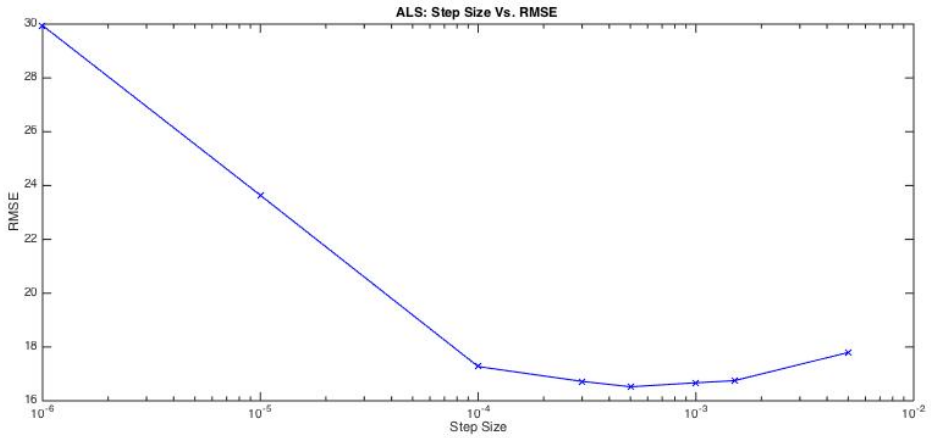

Fig 4: The plot shows the reduction in error as we varied γ, the step size of our model. For this test, the number of iterations was held constant at 10 iterations.

From fig 4 we can see that the minimum error occurred at $5 * 10^{-4}$. Using smaller step sizes would achieve the same or higher accuracy if given enough iterations but computation time would be significantly longer. Therefore we chose the step size of 0.0005. Step sizes larger than 10^{-2} caused overflow as we would experience overshoot issues.

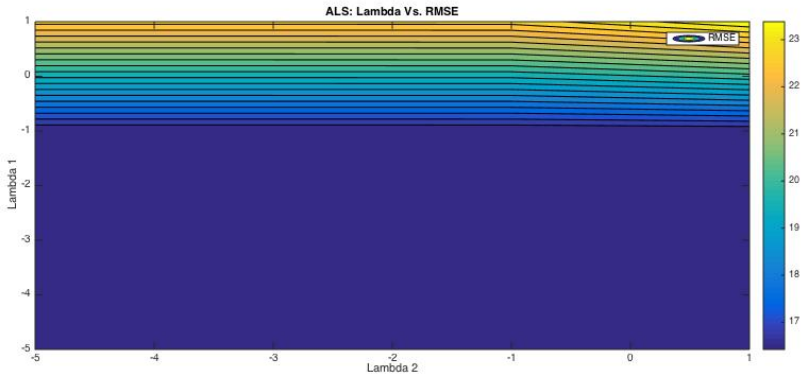

Fig 5: The contour plot shows how the RMSE error changed with $log_{10}(\lambda_1)$ and $log_{10}(\lambda_2)$. Orange corresponds to higher errors while blue corresponds to lower errors.

Fig. 5 shows that regularization of the tracks (λ_2) did not really affect our model while regularization of the users (λ_1) did have an effect. λ_1 values less than 0.1 all had similar errors. This corresponds to the blue plateau shown in fig. 5. Increasing λ_1 from 0.1 caused there to be a significant rise in error.

Cold Start Cases

The latent factor model predicts user(i)'s rating for track(j) as the inner product $U_i \cdot T_j$ (equation 1). However, the matrices U and T contain latent features for only the users and tracks included in the training set. The cold start cases occur when the test set includes users and tracks that are not part of the training set. Further details about the cases can be found in Appendix B.

When a prediction is to be made for a known user and artist but an unknown track, the latent features for the track are estimated as the average of the latent features of the known tracks by the same artist. However, if there are no tracks by the artist with known latent features, we estimate the latent features for the track as the average latent features of known tracks from the most similar artist. The most similar artist is the one whose artist profile is correlated the most (maximum inner-product) with the artist profile of the unknown artist. Similarly, the latent features for a new user are estimated as the latent features of the most similar known user. The user correlation is found by taking inner product of the new user profile with user profile of known users.

Design Decisions

Latent Factor Model-

In the milestone we discussed the possibility of using a kNN classifier to map users to the distinct peaks shown in fig. 1. However we chose to use the latent factor model instead because of its use in many recommender systems as well as its success in the kaggle competition [2,4].

Missing Data Entries-

Many of the entries in our user profile and artist rating data sets were missing. We originally chose to mark these entries with -1 so that they would be disregarded by our model. However, this required us to pass around a cell containing the indices of the missing features for each user. These indices were used by our functions to remove the -1's and rescale the feature vector accordingly. Later we found that setting these values to 0 cut the training and prediction time by a factor of 3 and reduced error by a few percent.

Passing indices to functions-

Since the matrices we were dealing with were so large, it would be inefficient to loop through all of the users or tracks when only a handful would actually be present. To mitigate this inefficiency, we passed the indices of interest to our function. To easily transfer between our training and test examples and our matrices M, U, and T we ordered users and tracks in terms of their id's. This way, given a user in our test set, we could easily find their user profile, latent features, or tracks they've rated by indexing their user id.

Optimization Method-

We tried using both stochastic gradient descent and ALS to update our latent features for tracks and users. We made the decision to go with ALS because it ran significantly faster than using stochastic gradient descent. Furthermore, stochastic gradient descent did not offer much reduction in error.

Latent Factor Initialization-

In order to run our ALS optimization, we needed to initialize U and T as well as determine the number of latent features to use in our model. We were unable to get good results initializing all vectors to ones. We found that initializing U to the user profiles significantly reduced the error that we saw from the initialization of 1's. Furthermore, we tried to initialize our matrix factors using an estimate from SVD. However, this was not effective because of the sparsity of M. Since our matrix was not completely filled in and SVD makes this assumption, we needed to initialize the unknown entries of M to the average value of our training set. This allowed us to use SVD to find matrix factors. However, since so many values of M were set to the average, a single eigenvector returned by the SVD function out scaled

the others by a factor of 1000. This large eigenvector caused numerical issues in the ALS optimization as well an increased training time as more iterations were needed to achieve the desired tolerance. The SVD initialization both increased both training time and error and was thus disbanded.

Results

Model Performance on Known Users and Tracks:

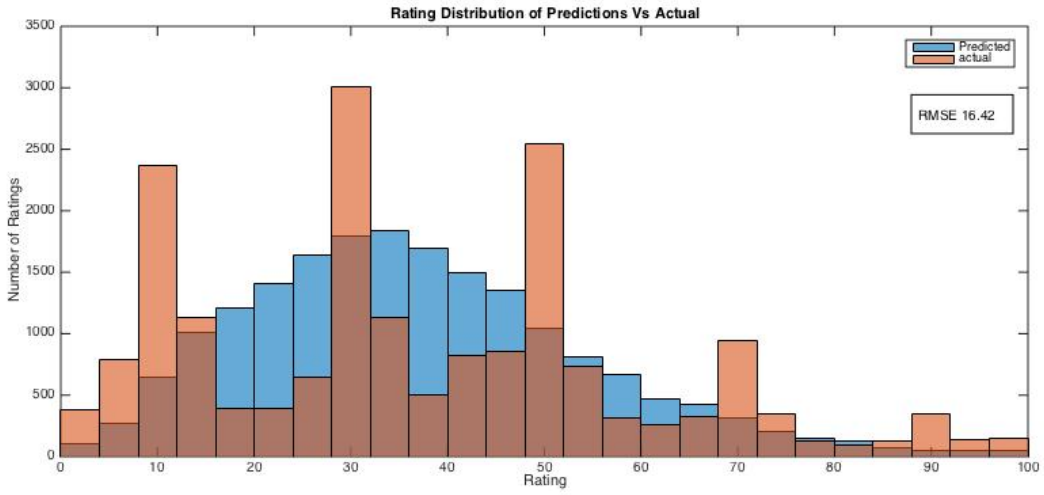

Fig 6: The histogram shows the distribution of predicted ratings and actual ratings for known users and tracks

We first tested our model's ability to predict ratings for new combinations of known users and tracks (fig. 6). For this test we randomized our training set and partitioned it into train and test folds with the ratio of 9:1. Randomizing the training data ensured a very low probability of encountering cold start cases. This allowed us to accurately test the performance of our matrix factorization model on a known set. We determined an RMSE of 16.42 for this cross validation. For this same test, our baseline average model achieved an RMSE of 21.47.

Model Performance on Cold Start Cases:

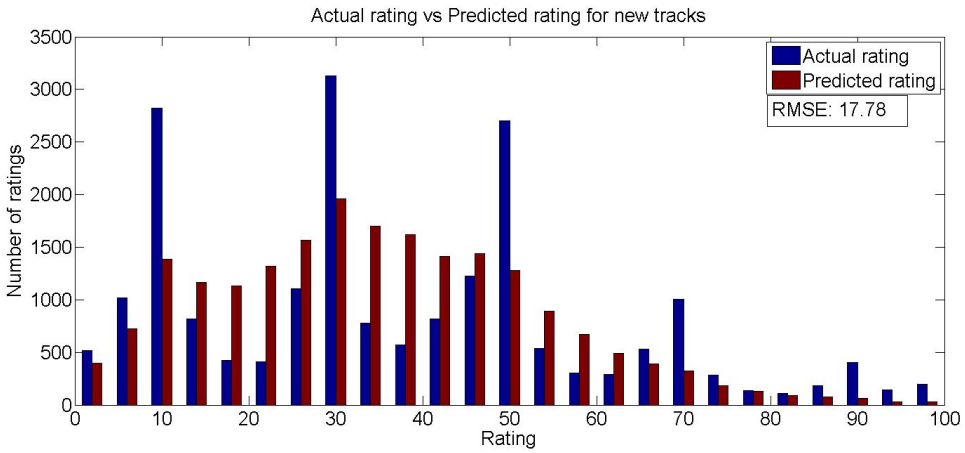

Fig. 7: Comparison of actual rating vs. predicted rating for new tracks

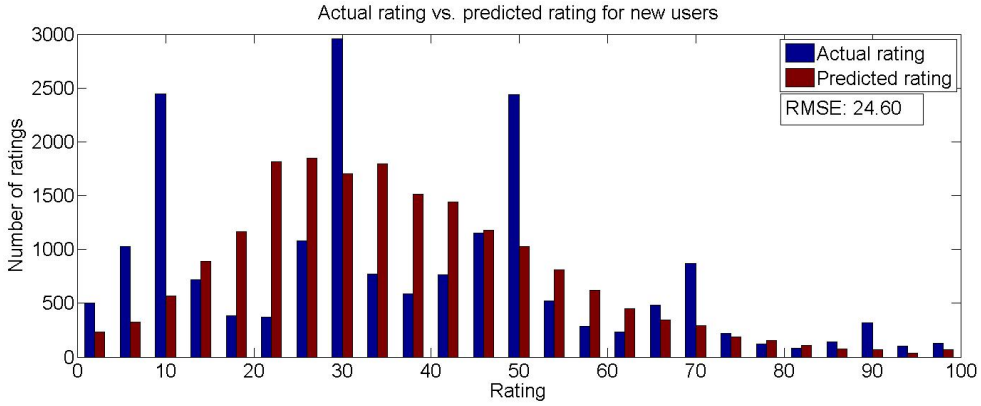

Fig. 8: Comparison of actual rating vs. predicted rating for new users

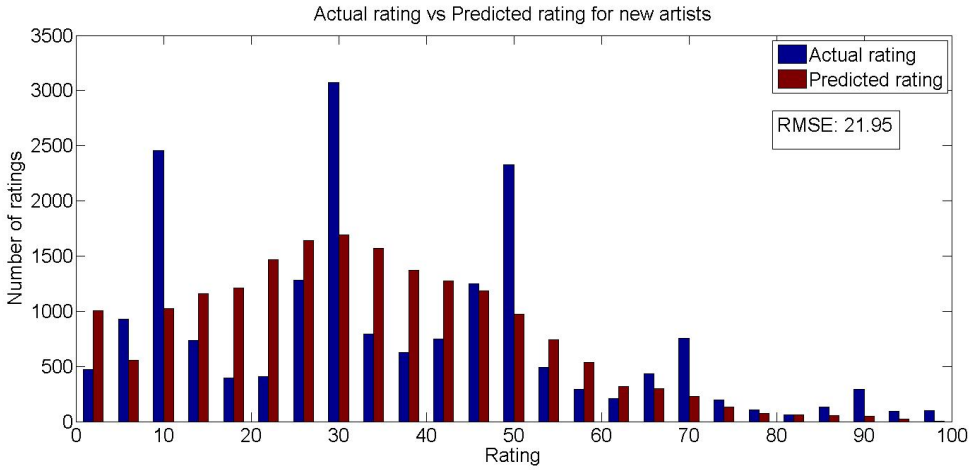

Fig 9: Comparison of actual rating vs. predicted rating for new artists

To evaluate our model's performance, we cross validated each of the cold start cases independently. Our cross validation could partition our training set into folds containing new artists, new tracks, or new users. To test the new track case, the training set included at least one track by all artists while the test set contained exclusively new tracks. To test the new user case, the training set included all artists and tracks while the test set included all new users. To test the new artist case, the test set contained exclusively new artists and tracks while users may have been known. The RMSE for the cold start case with new tracks, new users and new artists are 17.78 (Fig. 7), 24.60 (Fig. 8) and 21.95 (Fig. 9) respectively.

Model Performance in Causal System:

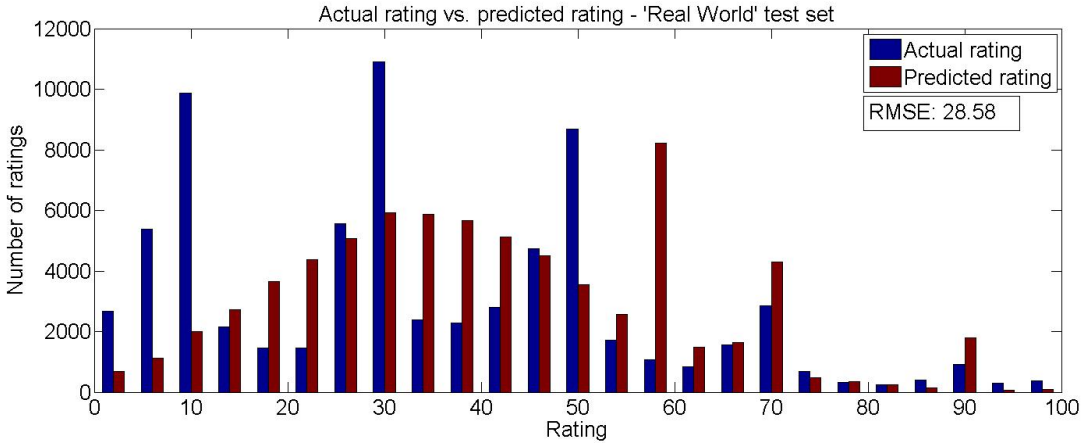

Fig. 10: Comparison of actual rating vs. predicted rating for new artists

For our causal system test, we divided our training and test sets based on time to model a real life situation where a recommender system needs to make predictions for unknown cases of users and/or items. We partitioned the user ratings such that the training set included ratings from the first 18 months and the train set included ratings from the next 6 months. This ensures that the training set will contain a combination of unknown users, tracks and artists. The RMSE for this case is 28.58 (Fig. 10). When we assigned the mode of our training set, 30, we found an RMSE of 22.18.

Discussion

Our model performed with the least error when given known users and tracks. The reason being that the latent features we optimized for the training set could be directly applied to the test set. While we were able to achieve low errors on the training set, our test set has higher errors because users may be rating new tracks that are completely unlike any tracks the user rated in our training set. The user's latent features may not predict so accurately when matched with a new set of latent features.

While our latent factor model outperformed the baseline average model, the predicted rating distribution still failed to resolve the peaks shown in fig 1. Instead, the distribution more closely followed the shape of a normal distribution (fig 6). This indicates that our model is still predicting an average rating of sorts instead of modeling the trends of how users actually rate tracks.

For the cold start cases we tested we found that the RMSE increased from that of the test with known users and tracks.This is expected because we must make the assumption that users and artists with similar profiles would rate and be rated similarly. The new tracks cold start case had the minimal cold start error. In this case, the latent features for the new track were derived by averaging latent features of known tracks by the same artist. Since we already have a history of the artist ratings, it is expected that this case would perform better than the other cold start cases. Our model assumes that new tracks will be similar or an "average" of tracks previously composed for the artist. Because time was not taken into account for the cold start cross validations, tracks recorded at similar times are contained in both the training and test set. Thus, it would make sense why taking the average track latent features for an artist in the training set would yield a small error.

The error in the new user and new artist cases is higher because our model tries to estimate the latent features of users and tracks by finding a similar user and artists. Our model is making the assumption that artists and users with similar profiles will be described by similar latent features. Since there are many factors that cannot be described by these profiles, it makes sense that this approach may not produce the best results.

The maximal RMSE error of our model was found in the causal time test. Here we introduced the concept of time into our training and test sets. The RMSE of our model for this test was 28.58. Predicting the mode would have yielded a RMSE of 22.18. The reason our model performed so poorly was that causality was introduced. The cold start case tests we performed were non-causal meaning that the data was partitioned with disregard for time. We may have trained on examples that occurred at a later time than the examples in our test set. Thus, our training set was a better representation of our test set. For the time-based test, the training and test sets may have been very different as users' preferences and artists' music changes over time. There is no way for our algorithm to determine how these users and artists would change with time. However, one thing that does not seem to change is the mode rating for tracks. While our model makes worse and worse predictions with time, the mode stays the same. Thus, in a causal system, assigning mode or mean ratings may be a better strategy than using a complex model that was trained on past ratings.

Sources of Error:

The largest source of error is that our model is based on finding averages. When we perform our ALS optimization, we are not finding latent features that predict exactly how a user rates, but latent features that minimize the error in the prediction. Rather than overfitting the data the algorithm settles on latent features that will give a good estimate, an average of sorts. This is reflected in the gaussian shaped curve seen in fig. 6. Since the data is clearly not distributed as a single normal distribution (fig 1), but a sum of several normal distributions our model can only perform so well. Our model is essentially trying to fit a single normal distribution to a sum of normal distributions. Thus, there is a significant amount of error inherent to our model selection.

Another large source of error is that we are trying to predict a track rating, but are given no information about the track. In order to make a better prediction about how a user would rate a track, we would need features that describe the tracks. Without these features, we must assume that new tracks can be described as averages of old tracks. This assumption is not always true and thus accounts for noise.

Some other sources of error include that this is a real world problem and many factors go into a user's rating other than their profiles and past histories. Our data sets contain many missing values and there are cases of users without user profiles to which we assigned the average of all users' features to the missing user.

Future Work:

In the cold start case with an unknown user, we assign the latent features of the most similar known user. Similarly, when trying to predict for a new artist, we first find a similar known artist and then use the average features of the tracks by the artist. In both these cases, we essentially use just one user or artist to get the needed features. This could be possibly be improved by using a kNN approach to find k similar users/artists to estimate the unknown latent features.

Our model derives an artists profile by averaging all the user's description of an artist. The artist profile is then used to find similar artists. This approach gives equal weight to each of the features in artist's profile. However, there could be some unique features within artists that might be more useful to relating artists. A latent factor matrix factorization model could learn the features that are most correlated among similarly rated artist.

An alternative approach to estimating the prediction could be to use classification. We can infer from fig. 1 that there are distinct peaks at ratings of 30, 50, 70, and 90. Rather than predicting a discrete value in the range of 0-100 like our model does, the classifier would assign one of the four rating classes to each data point. Such a classifier would fail to predict values to the left and right of each peak as observed in fig. 1. To improve this model, we could use a mixture of classification and regression in which the classifier predicts one of the four classes, and the regression model predicts the distribution around each class.

Works cited

1. "EMI Music Data Science Hackathon - July 21st - 24 Hours." *Description* -. N.p., n.d. Web. 15 Mar. 2015.

2. "Shanda Innovations 1st Prize Winner – EMI Music Data Science Hackathon." *Data Science London RSS*. N.p., n.d. Web. 15 Mar. 2015.

3. Koren, Yehuda, Robert Bell, and Chris Volinsky. "MATRIX FACTORIZATION TECHNIQUES FOR RECOMMENDER SYSTEMS." *IEEE Computer Society*(2009): n. pag. Web.

4. "EMI Music Data Science Hackathon - July 21st - 24 Hours." *Description* -. N.p., n.d. Web. 15 Mar. 2015.

Appendix A:

Table 1: Data After Preprocessing

Data field	Description	Type
Artist_id	unique identifier for artist	used for indexing
Track_id	unique identifier for track	used for indexing
User_id	unique identifier for user	used for indexing
Track Rating	User's rating of track	Integer [0,100]
Time_id	month data was taken	24 integer values
Gender	Male/Female	2 binary features
Age	Age of user	5 binary features corresponding to 20 year spans
Working	Employment status of user	9 binary features
Region	Where the user is from	4 binary features
List_Own	Hours spent listening to owned music	12 binary features
List_Back	Hours spent listening to music in background	12 binary features
Q_xx	19 questions about a user's music preferences	normalized value [0,1]
Heard_of	Has the user heard of the artist	5 binary features
Own-Artist_Music	Does the user own any of	5 binary features

	the artists music	
Like_Artist	User's rating of artist	normalized value [0,1]
Words_xx	82 words used to describe the artist	82 binary features

Table 2: Classification of Non-Numerical Values

Data Field	File	Original Data	Number of Binary Features
Heard_of?	Words.csv	• Heard of • Never heard of • Heard of and listened to music recently • Heard of and listened to music RECENTLY	5
Own_Artist_Music?	Words.csv	• DonÍt know • Own none of their music • Own a little of their music • Own a lot of their music • Own all or most of their music	5
Gender	users.csv	• Male • Female	2
Working	users.csv	• Employed 30+ hours a week • Employed 8-29 hours per week • Full-time housewife /househusband	5

		• In unpaid employment • Other	
Importance of Music?	users.csv	• Music means a lot to me and is a passion of mine • Music is important to me but not necessarily more important • I like music but it does not feature heavily in my life • Music has no particular interest for me	4

Appendix B

Cold Start Cases		
Unknowns	**Knowns**	**Solution**
Track	Artist User	• Estimate the latent features of the track as the mean of known tracks by the same artist
User	Track Artist	• Estimate the latent features of the unknown user as the latent feature of the most similar known user
Track User	Artist	• Estimate the latent features of the track as the mean of known tracks by the same artist • Estimate the latent features of the unknown user as the latent feature of the most similar known user
Track Artist	User	• Estimate the latent features of the track as the average of latent features of the track of the most similar artist
Track Artist User	None	• Estimate the latent feature of the unknown user as the latent features of the most similar known user • Estimate the latent feature of the track as the average of the latent features of the track of the most similar artist