Predicting Billboard Top 100 Songs

Julien Kawawa-Beaudan and Gabriel Garza
julienkb and ggarza5 @ stanford.edu

December 11, 2015

1 Introduction

The goal of this project was to create a machine learning project which could successfully predict whether a song could achieve mainstream commercial success. The motivation for the project was that we are both interested in music, and were curious whether a machine could successfully predict something that seems to be based on purely subjective human judgment.

2 Dataset

For this project, we used a 10,000 song subset of the publicly available Million Song Dataset. The Million Song Dataset, created through by Columbia University’s LabROSA and The Echo Nest, contains data about a million songs sampled from many music genres, time periods, and places. However, due to the size of full dataset, we opted to use the smaller 10,000 song subset.

2.1 Labels

Positive: Songs were labeled as positive if the song had appeared on a Billboard Top 100 chart from any month since 1958, when the Billboard Top 100 was created.

Negative: Songs were labeled as negative if the song’s artist had never appeared on a Billboard Top 100 list. The rationale for this labeling was that there were many songs in our dataset by hit artists, which might have been too similar to that artists’ hit songs for any classifier to distinguish.

Our final dataset included 254 positive examples, 7302 negative examples, and 2444 non-hit songs by hit artists (which we excluded).

2.2 Features

The Million Song Dataset contains both metadata and audio data for each song. In fact, one of our main challenges was deciding which features to use, since each song had more data than we could reasonably train on if we included all the audio data. The features included genre labels, metadata, and 12 x n arrays for the pitches, timbres, and loudness at each “segment”, which typically represents the beginning of a new note [4].
Genre: Each song was hand-labeled with many genres by the creators of the Million Song Dataset. We collected the six most common labels, and then created a feature vector for each song where each value was an indicator for whether that label was applied to the song.

Pitches and timbres: Each song contains a 12 x n array containing the pitches at each time slice in the song. In addition, each song also includes two 12 x n arrays containing a corresponding numerical value for the timbre and loudness of those pitches.

Metadata: The database contains metadata for each song. We chose to keep 11 of the features from the metadata, including song duration, musical key, mode (major or minor), tempo, year of release, and time signature. We also chose to include the artist and song “hottness” features, a metric created by The Echo Nest which represents the popularity of the song when the database was created. Originally, we hoped to use the metadata, including the hotness feature, as a control or baseline feature to compare our models trained on other feature sets against.

Since we were unsure which features would be the most helpful, we ran all the models with several different sets of features: just the genre label, just the metadata, the pitches and timbres from 4 equally spaced points in the song, the pitches and timbres for 10 time slices at 30 and 60 seconds into the song, and finally a combined set using genre, metadata, and the audio data at 60 seconds.

3 Machine Learning Algorithms

The first classifiers we experimented with were simple linear regression and logistic regression. However, because our data was very imbalanced (only 3.3% positive), both of our regression models performed poorly. So, we also experimented with increasing the penalty for misclassified positive examples, as suggested in a paper we discovered while researching training models on imbalanced datasets. [3].

We also ran other standard machine learning algorithms, which were Naïve Bayes, Support Vector Machines, Gaussian Discriminant Analysis, and Decision Trees.

4 Results and Discussion

One of our major challenges was that our dataset was very unbalanced, which made it very simple to achieve a highly accurate classifier simply by always predicting false. In particular, SVM always labeled all our data as negative, which is why it is omitted from the following figures. Although it was not extremely surprising that there was no clear decision boundary when trained on a linear kernel (since there was not a “stand-out” feature that we intuitively believed would separate hit songs compared from non-hit songs), it was a little surprising that an SVM trained on a Gaussian kernel also returned the same results.

Because high accuracy was easy to achieve without creating a meaningful classifier, we choose to measure classifiers by the positive recall and precision.
rates instead. The results of training on all the combinations of features and models are shown in the graphs below.

The low precision but very high recall for both linear and logistic regression illustrates the reverse problems of the SVM - labeling almost all the points as positive, which is equally unhelpful. In order to fix this problem, we also experimented with an implementation of stochastic gradient descent for logistic regression, in which we penalized misclassified positive examples more than misclassified negative examples. In practice, however, this did not improve the overall positive precision of the classifier; the weight only changed the overall fraction of predictions which were positive, but without actually making improving the positive precision. The values shown in the chart to the left are for the weighted regression when the penalty for misclassified positives is 10 times that for misclassified negatives, but in practice the weight had little effect on the positive precision.

One surprise we had when running the experiments was that metadata, which included an artist "hotttness" and song "hotttness" feature, did not significantly outperform the other feature sets. Our original intention was to use the metadata feature set as a baseline to compare other feature sets on, but in practice only the Naive Bayes and Gaussian Discriminant analysis classified more accurately on the metadata than on the actual audio.

Another interesting result of these experiments was that the Naive Bayes classifier performed roughly the same on the random audio - the pitches and timbres from 4 equally spaced points in the song - as it did on the sequence of audio at the 30 second and 60 second marks. The two other useful classifiers, Gaussian discriminant analysis and decision trees, significantly improved when using the sequential audio instead of the random audio, which matches the human intuition that a human could recognize a song given a 3-second clip of a
song, but not based on several random moments in a song. However, this makes sense in some respects, because the positive precision for Naive Bayes on all the audio features is fairly low, about 6%, which is only two times more accurate than randomly guessing. Since the critical assumption with Naive Bayes is that the features are independent - which is clearly not the case for audio features of music - this could mean that the independence assumption is equally bad for all sets of audio features.

<table>
<thead>
<tr>
<th>Feature set</th>
<th>Algorithm</th>
<th>Accuracy</th>
<th>+ Recall</th>
<th>+ Prec.</th>
<th>F-Measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Metadata</td>
<td>GDA</td>
<td>0.9684</td>
<td>0.2194</td>
<td>0.75</td>
<td>0.339</td>
</tr>
<tr>
<td>Metadata</td>
<td>Decision Tree</td>
<td>0.9555</td>
<td>0.3283</td>
<td>0.3234</td>
<td>0.325</td>
</tr>
<tr>
<td>Audio, 30 sec</td>
<td>GDA</td>
<td>0.9605</td>
<td>0.1633</td>
<td>0.3048</td>
<td>0.213</td>
</tr>
<tr>
<td>Audio, 30 sec</td>
<td>Decision Tree</td>
<td>0.9517</td>
<td>0.3182</td>
<td>0.2864</td>
<td>0.301</td>
</tr>
<tr>
<td>Audio + Metadata</td>
<td>GDA</td>
<td>0.9587</td>
<td>0.1633</td>
<td>0.2759</td>
<td>0.205</td>
</tr>
<tr>
<td>Audio + Metadata</td>
<td>Naive Bayes</td>
<td>0.9687</td>
<td>0.096</td>
<td>0.6333</td>
<td>0.167</td>
</tr>
<tr>
<td>Audio + Metadata</td>
<td>Decision Tree</td>
<td>0.967</td>
<td>0.298</td>
<td>0.4504</td>
<td>0.359</td>
</tr>
</tbody>
</table>

Table 1: The results of our best classifiers on the testing data. Models trained on audio at 60 seconds performed very similarly to audio at 30 seconds.

One last observation about our results is that we were actually able to classify songs quite well using only the audio information, represented as an array of frequencies and pitches. This was quite surprising to us, because even the authors of the Million Song dataset reported that the audio information encoded in the dataset is not sufficient to accurately reconstruct the song. We assumed that because the song would not be clearly recognizable to a human, a computer would not be able to classify hit songs. In addition, these results are surprising considering that popular music today is very different from popular music when the Billboard Top 100 list first began in 1958. It seems very plausible that a hit song from the 90’s would not be successful at all in the 50’s. There are two possible explanations for the success of our classifiers despite this obstacle - either our models identified some universal features for detecting hit songs, or the models actually learned different “clusters” of features corresponding to different types of music popular throughout the years. Since the two algorithms which consistently outperformed the others were Gaussian discriminant analysis and decision trees, which are capable of capturing multiple “peaks” or modes, it seems much more likely that the second option is the case.

5 Conclusion

We successfully trained several classifiers to recognize Billboard Top 100 songs using data provided in the Million Song Dataset with significantly greater accuracy than randomly guessing or always guessing false. Overall, our best classifier based on recall and precision rate was a Gaussian discriminant model on the metadata features. However, we also had success classifying using the audio pitches and timbres data provided by the Million Song Dataset, using Gaussian discriminant analysis and decision trees.
6 Future Work

Although our project successfully classified hit songs using the data provided by the Million Song Dataset, there are many directions for future work. In particular, both academic papers [2] and past CS 229 projects [5] have focused on working with music in the MIDI format, which contains information about the instruments and actual notes being played, as opposed to an array of pitches being played. The MIDI format is cleaner and easier to analyze than a simple array of pitches, because many instrument sounds are combinations of different frequencies. For example, percussion sounds often span many frequencies, and make it difficult to analyze the main components of music when using an array of pitches, while the MIDI file would isolate the percussion track from other instruments.

In addition to making analysis of single instrument voices much easier, the MIDI format would also allow using time-series analysis of the music. A major flaw of representing music as an array of pitches at each time-slice is that this representation fails to capture the time variation of music. MIDI files, in contrast, would allow extraction of features such as rhythms, recurring note patterns, and harmonic structure.

Finally, another research possibility which seems promising is using lyrics to predict song success. The use of lyrics, either in as a bag-of-words or as n-grams of words, could be very interesting as a feature for predicting song success. However, we decided against using lyrics for this project because the task of collecting lyrics for each song would be time-consuming and also require us to remove many songs from our dataset which contain no lyrics or were sung in another language.

7 Acknowledgments

We would like to thank Prof. Andrew Ng and the CS229 Teaching Staff for teaching the Machine Learning course.

References


