Predicting Hit Songs with MIDI Musical Features

Keven (Kedao) Wang
Stanford University
kvw@stanford.edu

ABSTRACT

This paper predicts hit songs based on musical features from *MIDI* files. The task is modeled as a binary classification problem optimizing for precision, with *Billboard* ranking as labels. *Million Song Dataset (MSD)* is inspected audibly, visually, and with a logistic regression model. *MSD* features is determined too noisy for the task. *MIDI* files encodes pitch duration as separate instrument tracks, and is chosen over *MSD*. Fine-grained instrument, melody, and beats features are extracted. Language models of *n-grams* are used to transform raw musical features into word-document frequency matrices. *Logistic Regression* is chosen as the classifier, with increased probability cutoff to optimize for precision. An ensemble method that uses both instruments/ melody as well as beats features produces the peak precision 0.882 at probability cutoff 0.998 (recall is 0.279). Alternative models and applications are discussed.

Keywords

Music, Hit Song, Classification, MIDI

1. INTRODUCTION

The goal of this project is to predict hit songs based on musical features. A song is defined as a hit if it has ever reached top 10 position on a *Billboard* weekly ranking. Predicting hit songs is meaningful in numerous ways:

1. Help music stream services to surface upcoming hits for better user engagement
2. Improve the iteration process for artists before releasing music to the public

Various factors determine whether a song is popular/a hit, including the intrinsic quality of the music piece, and psychological/social factors [Pachet and Sony 2012]. The latter encompass peer pressure, public opinion, frequency of listening, and artists' reputation. These psychological/ behavioral factors are harder to quantify, and is out of scope for this project. The popularity of a song does correlate to its intrinsic quality [Pachet and Sony 2012]. Therefore this paper focuses on the analysis of musical features, which are quantifiable and encapsulated in the audio file itself. The following musical features are analyzed.

1. Timbre/instrument
2. Melody
3. Beats

Relatively few projects have explored the hit song prediction space [Pachet and Sony 2012, Ni et al. 2011, Dhanaraj and Logan 2005, HERREMANS et al. 2014, Fan and Casey 2013, Monterola et al. 2009]. A majority of the research have taken low level features from audio file formats such as .mp3, and .wav by using signal processing techniques to extract *MFCC* values spanning short time window [Serrà et al. 2012, Pachet and Roy 2009]. This project extract instrument, melody and beats features from *MIDI* files. Surprisingly good results are obtained in this project.

Since it is more valuable to correctly predict popular songs than correctly predicting unpopular songs, the precision metric is optimized.

2. DATA

Both *MIDI* and *Million Song Dataset* are explored for feature extraction. *MIDI* is shown to produce much higher quality features that results in higher performance, and is therefore chosen for this project.

2.1 MIDI

Musical Instrument Digital Interface (MIDI) is a technical standard that allows musical devices to communicate with each other. A *MIDI* file contains up to 16 tracks, each representing an instrument. Each track contains messages, which encodes the pitch and duration of an instrument key press.

MIDI files are close approximations of original music piece. Although lacking the fidelity and human-ness of raw audio files such as .mp3 and .wav, it is nevertheless a faithful representation of high level musical features of timbre, melody, and beats. *MIDI* suffices the purpose of feature extraction in this paper.

1752 *MIDI* songs are used as training samples, with exactly 50-50 split between positive and negative training examples. The definition of positive and negative labels is outlined below in "Labels" section.

2.2 Million Song Dataset

The *Million Song Dataset (MSD)* is a musical feature dataset of one million contemporary songs. It is publicly available from the joint collaboration between *LabROSA* and the *Echo Nest* [LabROSA 2014]. The dataset is readily parseable via a Python API with getters and setters to individual features. Further information could be queried via the Echo Nest API, which is freely available [the EchoNest 2014].

In pursuing this project, visual and audio examination, as well as a *Logistic Regression* model was used to test the effectiveness of the 10K subset. However, the result was unsatisfactory. The following introduces inspections and findings on why *MSD* is not ideal for this project.

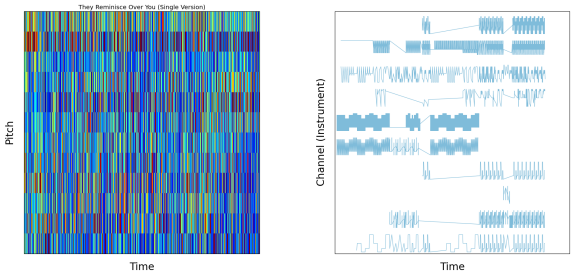

Figure 1: Left: *Million Song Dataset*: Does not distinguish between tracks. Right: *MIDI*: Represents each instrument as separated tracks.

In *MSD*, the features of interest are:

- **Popularity labels**: *"Hotttness"* score, a score from 0 to 1.0 from the Echo Nest representing popularity.
- **instruments/melody features**: *"Pitch"* matrix of size 12 by length-of-song. Size 12 represents the 12 semitones in an octave. The value of each cell correlates to the relative strength of particular semitone at a given time. Sampling is done every 250 milliseconds, with some variation based on the beats.

The *pitch* matrices are not effective representation of instruments/melody features, as it does not distinguish between instruments. Percussion can greatly distort the dominant melody of the song [Jiang et al. 2011]. Figure 1 is a visualization comparison between *MSD* and *MIDI* melody features. In the left figure, the red color represents a "loud" signal and the blue color represents a "quiet" signal. The visualization shows a noisy representation of melody. A listening test was performed on the transformed sine wave, with frequency determined by the "loudest" semitone in a pitch matrix column. The result audio is completely unrecognizable for multi-instruments/track songs. A baseline model is implemented, with 10-fold cross validation showing a fluctuation around 50% precision, recall, and f1 score. This is not an improvement over random baseline.

MIDI files, on the other hand, encodes each instrument into separate tracks as in the right figure. Therefore it is decided that the *MSD* dataset is not effective as melodic features, and is not used for this project.

2.3 Labels

The problem of hit song prediction is modeled as a binary classification problem, with positive labels representing the popular songs and negative labels representing unpopular ones. The *Billboard* ranking is used to determine whether a song is popular. *Billboard* is a prominent music popularity ranking based on radio plays, music streaming and sales published weekly. *Billboard* contains ranking dating back to the 1950s. In this paper, the labels are assigned as follows:

- **Positive**: if a song ever reached the top 10 position on any *Billboard* ranking (since 1950)
- **Negative**: if a song's artist never had any song reaching top 100 position on any *Billboard* ranking

This requirement is rather strong, leaving out a large mid-class songs in between. This is done to emphasize the differences between the two classes.

3. PREPROCESSING

3.1 Type 0 to Type 1

Two types of *MIDI* music files are of interest. *Type 1* is easier to process for feature extraction, as each track represents distinct instruments. Therefore all *type 0 MIDI* files are converted into *type 1*.

1. **Type 1**: each individual track represents one musical instrument. (80% of training samples)
2. **Type 0**: one single track contains messages across all channels. (20% of training samples)

The Open-Source Python module *MIDO* provides a friendly API that parses a *MIDI* file into native python data structures [Bjørndalen and Binkys 2014]. It is used to extract instruments/melody and beats features from *MIDI* files.

4. FEATURE EXTRACTION

Fine-grained features are engineered to capture the subtle characteristics of a song. Language models of *n-grams* are used to capture the building blocks of melody and beats. In order to extract the features, the following *MIDI* messages are of interest:

- `Set tempo`: specifies tempo of music piece in microseconds per quarter note (beat)
- `Note on`: specifies the start of a music note event (e.g. piano keyboard press)
  - `Note`: specifies the pitch, with 60 representing the middle C
  - `Velocity`: specifies how much force a note is played with (a note on message with velocity 0 is the same as a note off message)
- `Note off`: specifies the end of a music note event
- `Program change`: specifies the instrument for track
- `Delta time`: specifies the number of ticks since last *MIDI* event, for each *MIDI* event. This can be converted to delta time in seconds.

Besides, the metadata of `Pulse per Quarter-Note` is needed. It specifies number of ticks per beat. This is needed to compute time delta between *MIDI* messages.

4.1 Instruments

Each *MIDI* track contains a program change message, with instrument type encoded with 0 - 127. A manual grouping on instrument types is done based on suggestions from [McKay 2004] and [Association 2014]. The grouping shrinks the feature space, while capturing the distinguishing timbre of instrument classes.

Table 1: *MIDI* instrument grouping by program change number (0 - 127)

MIDI Program Change Number	*Instrument*
0 - 4	Keyboard
5 - 6, 17, 19	Electric
7, 8,	Other
9 - 16	Chromatic percussion
18 - 24	Organ
25, 26	Acoustic guitar
27 - 32	Electric guitar
33 - 40	Bass
...	
113 - 120	Percussive

4.2 Melody

Melody features are represented by chord progression characteristics. These defining characteristics in music theory is captured in features below.

- **Consecutive two notes** (*2-grams*): captures musical interval. Musical interval defines transition between two consecutive music notes. [Wikipedia 2014].
- **Consecutive three notes** (*3-grams*): [Caswell and Ji 2013] suggests that Markov models looking at previous 3 or 4 note pitches produced the best results.

The instruments and melody features are combined to represent the distinct instruments/melody combination. As an example, the following *MIDI* note on messages are transformed into the following features.

MIDI messages:

```
<track 1>
program_change, value = 0 # instrument:  piano
note_on, note = 60, velocity = 64
note_on, note = 62, velocity = 64
note_on, note = 67, velocity = 64
<track 2>
program_change, value = 27 # instrument:  electric guitar
note_on, note = 72, velocity = 64
note_on, note = 76, velocity = 64
```

Extracted features:

```
2-grams:  keyboard:60:62, keyboard:62:67, electric_guitar:72:76
3-grams:  keyboard:60:62:67
```

4.3 Beats

The beats features are represented by extracting time delta between consecutive musical notes. The idea of *2-grams* and *3-grams* is used here again. In *MIDI*, time delta between *MIDI* messages is specified in number of ticks. The following calculation is needed to calculate the delta time (in seconds) between consecutive *MIDI* messages:

$$delta\_time = delta\_ticks \cdot \frac{tempo}{1000000.0 \cdot PPQN} \tag{1}$$

tempo is in microseconds per quarter note.
PPQN (Pulse Per Quarter Note) is in ticks per quarter note.

4.3.1 Note duration

MIDI contains various messages other than note on. Time delta occurs between each consecutive *MIDI* messages. Work is done to accumulate the time delta between consecutive note on messages. Afterwards, time delta is converted to *beat-per-minute*, a standard way of capturing tempo information and accounting for small time delta.

4.3.2 Percussion

Track 9 is always the percussion track. Percussion track is more important as a beats feature than other instrument tracks. Therefore a prefix is added for features extracted from percussion track to distinguish from other tracks.

4.4 Dimensionality Reduction

Two dimensionality reduction techniques are explored to avoid overfitting. However, due to their limitations and the effectiveness of regularization term in Logistic Regression, no dimensionality reduction algorithm is used in this project.

PCA (*Principal Component Analysis*) transforms feature space into a lower-dimensional subspace composed of pairwise linearly independent vectors. In practice PCA tends to favor features with high variance. *Feature Selection* prioritizes features with highest marginal increase/decrease in performance. *Feature Selection* itself is time consuming.

5. MODELS

The problem is modeled as a binary classification problem. This allows for plug-and-play of many off-the-shelf classification algorithms. A more practical model is outlier detection, since it is much more valuable to predict a popular song than an unpopular song. However, I was not able to find such a model with satisfying results. The following models are used in training and testing on the dataset.

5.1 Logistic Regression

Logistic Regression is chosen as the model for task. Logistic Regression outputs the confidence probability for each prediction. This is ideal for optimizing for precision, since the probability cutoff for positive labels can be increased to form a "stricter" criteria on popular songs. A regularization coefficient λ is added and iterated on to decrease overfitting.

$$J(\theta) = \frac{1}{2m}[\sum_{i=1}^{m}((h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^{n}\theta_j^2] \tag{2}$$

$$h_\theta(x) = P(y = 1|x;\theta) = \frac{1}{1 + e^{-\theta^T x}} \tag{3}$$

5.2 SVM

SVM is regarded as one of the best "off-the-shelf" models. It has the advantages of being time-efficient and avoiding overfitting. In this project, time-efficiency is not a concern given the relatively small sample size (1700+) to work with.

$$y_i(w \cdot x_i - b) \geq 1 - \zeta_i \tag{4}$$

$$\underset{w,\zeta,b}{\operatorname{argmin}} \left\{ \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\zeta_i \right\} \tag{5}$$

$$y_i(w \cdot x - b) \geq 1 - \xi_i, \xi_i \geq 0 \tag{6}$$

5.3 Naive Bayes

Naive Bayes assumes each feature is independently distributed. This assumption is too strong for melody segment features (*n-grams*), as they are dependent on neighbors and the overall chord progression distribution of the song.

$$\underset{y}{\operatorname{argmin}} \left\{ P(y = y_i)\prod_{i=1}^{n} P(x_i|y) \right\} \tag{7}$$

5.4 One-Class SVM

The supervised outlier detection problem is attempted. *One-Class SVM* is an one class classification algorithm that takes only positive training examples and adds a negative example at the origin. Compared to other clustering (*K-means*) and outlier detection algorithms (*Mixtures of Gaussian*), *One-Class SVM* allows for supervised learning.

6. RESULTS

6.1 Models Comparison

Logistic Regression (cutoff probability = 0.5, regularization $\lambda = 1.0$), *SVM*, *Naive Bayes*, and *One-Class SVM* are used to compare performance. 10-fold cross-validation is performed on the 1752 samples (50% positive, 50% negative). Mean precision is used as evaluation criteria. *Logistic Regression* and *SVM* resulted in the highest mean precision.

Table 2: Logistic Regression and SVM produces the highest mean precision

features	*Model*	*Precision*	*Recall*	*F1*
Beats	1-Class SVM	0.488	0.406	0.439
	Naive Bayes	0.577	0.647	0.608
	Logistic Reg.	0.600	0.618	0.607
	SVM C=1.0	**0.625**	0.575	0.597
Melody + Instrument	1-Class SVM	0.514	0.491	0.501
	Naive Bayes	0.559	0.628	0.590
	Logistic Reg.	**0.596**	0.633	0.612
	SVM C=1.0	0.595	0.608	0.601

6.2 Ensemble Method

The ensemble method with *Logistic Regression* (regularization $\lambda = 1.0$) gave the highest overall precision. This method uses both instruments/melody features and beats features. Two separate *Logistic Regression* classifiers are run on each feature sets separately. The combined output is positive if both predicted probabilities are greater than a confidence cutoff.

Precision is optimized by increasing the probability cutoff. The **best precision 0.882** is achieved at probability cutoff of 0.998 (recall is 0.279). The increase in probability cutoff can be intuitively understood as the increased confidence required to qualify for a positive label (a popular song).

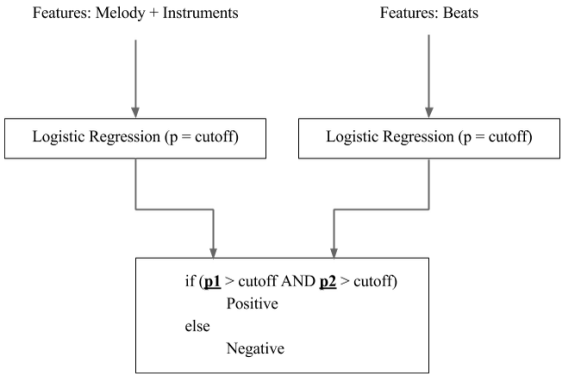

Figure 2: Ensemble method produces the best performance by combining both instruments/melody features and Beats.

6.3 Features Comparison

Using only instruments/melody features or only beats features resulted in identical performance (roughly 60% precision). Combining the two features as in ensemble method improves the precision by roughly 15% or more.

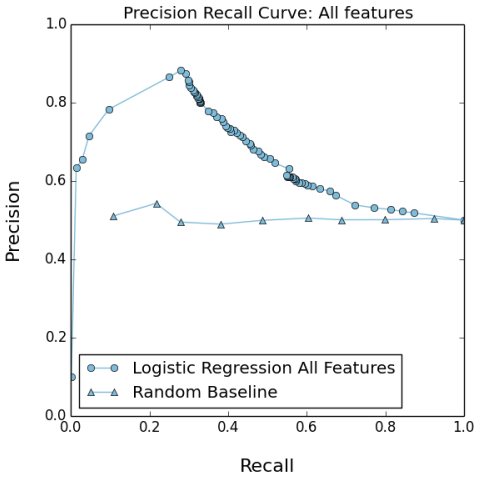

Figure 3: Using the ensemble method, precision peaks at 0.882. Recall is 0.279.

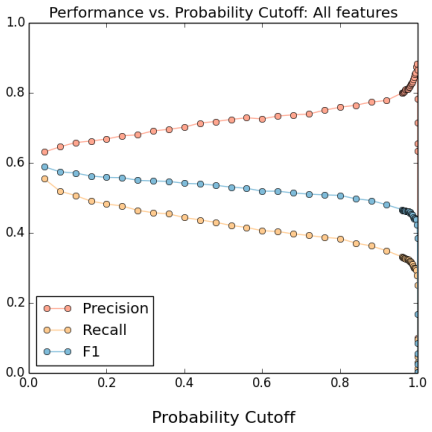

Figure 4: Using the ensemble method, probability cutoff can be increased to increase precision.

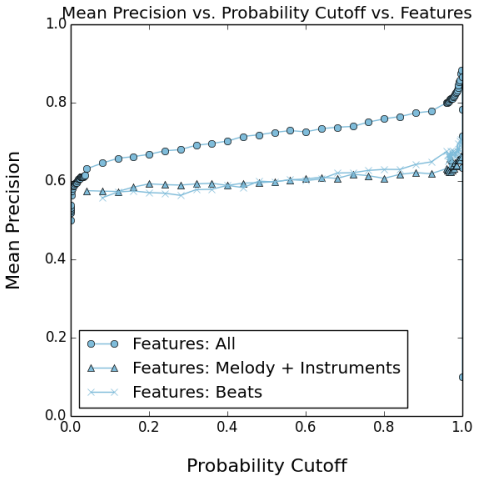

Figure 5: Combining instruments/melody and beats features results in 15%+ increase for mean precision.

6.4 Regularization

Experiment is done in tweaking the regularization parameter λ in *Logistic Regression* ($p = 0.5$). The result shows an insignificant change in precision as a result of regularization parameter change. Therefore λ is kept at the default 1.0.

7. DISCUSSION

The peak precision 0.882 at probability cutoff 0.998 is surprisingly good. The corresponding recall is 0.279. Precision is optimized since it is more valuable to have true positives than true negatives. The high precision demonstrates that:

1. The distinguishing characteristics between popular and unpopular songs can be learned.
2. *MIDI* files are able to produce high quality features for hit song prediction purposes.
3. It is promising to borrow language models for melody and beats feature extraction.
4. The feature extraction of instruments, melody, and beats features is able to capture the distinguishing characteristics of a song. The ensemble of instruments/ melody and beats features is promising.

The top 100 features with highest *Logistic Regression* coefficients are analyzed. There is no clear pattern on these features. Given the large feature space and the large number of features a single song has, a positive prediction is likely attributed by the aggregation of a large number of features.

One-Class SVM hardly gives any improvement over random baseline. This can be explained by the strict information loss by replacing all negative training examples with a single negative example at origin.

8. FUTURE WORK

Supervised outlier detection models could be explored. In this project, the hit song prediction problem is modeled as a binary classification problem. Modeling as outlier detection would be more suitable because:

1. It is more valuable to correctly predict popular songs than predicting negative ones. Modeling as outlier detection removes the need to collect negative label songs, which are vast and harder to define.
2. There could be a vast number of reasons as to why a song is not popular. It is better to focus on finding defining features of popular songs.

Neural Network has produced excellent results for speech recognition tasks, and could be used for feature selection purposes. Generative models such as *Mixture of Gaussians* and *K-means* can be used for outlier detection. *Neural Networks* could be supervised to learn features for these unsupervised models.

A natural next step would be auto hit song composition. The result could augment human in composing hit songs by offering inspirations. A combiner is needed to reconstruct a song coherently from building blocks of melody and beats. The bottom-up construction mechanism would first combine *n-grams* into musical bars, then into verses/choruses, and eventually into the entire song.

9. REFERENCES

[Association 2014] MIDI Manufacturers Association. 2014. General MIDI Level 1 Sound Set. (2014). http://www.midi.org/techspecs/gm1sound.php

[Bjørndalen and Binkys 2014] Ole Martin Bjørndalen and Rapolas Binkys. 2014. Mido - MIDI Objects for Python. (2014). http://mido.readthedocs.org/en/latest/

[Camenzind and Goel 2013] Tom Camenzind and Shubham Goel. 2013. #jazz : Automatic Music Genre Detection. (2013).

[Caswell and Ji 2013] Isaac Caswell and Erika Ji. 2013. Analysis and Clustering of Musical Compositions using Melody-based Features. (2013).

[Dhanaraj and Logan 2005] Ruth Dhanaraj and Beth Logan. 2005. Automatic Prediction of Hit Songs.. In *ISMIR*. 488–491.

[Fan and Casey 2013] Jianyu Fan and Michael A Casey. 2013. Study of Chinese and UK Hit Songs Prediction. (2013).

[HERREMANS et al. 2014] Dorien HERREMANS, David MARTENS, and Kenneth SÖRENSEN. 2014. *Dance hit song prediction*. Technical Report.

[Jiang et al. 2011] Nanzhu Jiang, Peter Grosche, Verena Konz, and Meinard Müller. 2011. Analyzing chroma feature types for automated chord recognition. In *Audio Engineering Society Conference: 42nd International Conference: Semantic Audio*. Audio Engineering Society.

[LabROSA 2014] Columbia LabROSA. 2014. Million Song Dataset. (2014). http://labrosa.ee.columbia.edu/millionsong/

[McKay 2004] Cory McKay. 2004. *Automatic genre classification of MIDI recordings*. Ph.D. Dissertation. McGill University.

[Monterola et al. 2009] Christopher Monterola, Cheryl Abundo, Jeric Tugaff, and Lorcel Ericka Venturina. 2009. Prediction of potential hit song and musical genre using artificial neural networks. *International Journal of Modern Physics C* 20, 11 (2009), 1697–1718.

[Ni et al. 2011] Yizhao Ni, Raúl Santos-Rodríguez, Matt Mcvicar, and Tijl De Bie. 2011. Hit song science once again a science? (2011).

[Pachet and Roy 2009] François Pachet and Pierre Roy. 2009. Analytical features: a knowledge-based approach to audio feature generation. *EURASIP Journal on Audio, Speech, and Music Processing* 2009 (2009), 1.

[Pachet and Sony 2012] François Pachet and CSL Sony. 2012. Hit song science. *Music Data Mining* (2012), 305–26.

[Serrà et al. 2012] Joan Serrà, Álvaro Corral, Marián Boguñá, Martín Haro, and Josep Ll Arcos. 2012. Measuring the evolution of contemporary western popular music. *Scientific reports* 2 (2012).

[the EchoNest 2014] the EchoNest. 2014. Echo Nest API Overview. (2014). http://developer.echonest.com/docs/v4

[Tzanetakis and Cook 2002] George Tzanetakis and Perry Cook. 2002. Musical genre classification of audio signals. *Speech and Audio Processing, IEEE transactions on* 10, 5 (2002), 293–302.

[Wikipedia 2014] Wikipedia. 2014. Interval (music) — Wikipedia, The Free Encyclopedia. (2014). http://en.wikipedia.org/w/index.php?title=Interval_(music)&oldid=627299987