
For-Tune Teller

Gerard Touma
Mridul Krishnan
Sambhav Jain

Stanford University, Stanford, CA 94305 USA

gtouma@stanford.edu
mridulk@stanford.edu
sambhav@stanford.edu

ABSTRACT

In this paper, we document the development of a learning algorithm that rates the popularity of a song on a scale of 0 to 5, using a carefully chosen set of musical features. Along with personal interest, the main motivation is to be able to help both the artists and label/advertisement companies in the music industry. We considered two different datasets (MP3 and MSD) and we tried to mitigate the effect of unwanted factors as much as possible by carefully choosing the songs. We trained, tuned, and compared three different learning algorithms, and chose the SVM as it has the best performance. The results show that we can correctly classify a given song with 37.8% accuracy into the correct class, and in the event of misprediction, there is a 71% chance that the error is only one class away.

1. INTRODUCTION

Although music is subjective in its own terms, the quality and success of a song is far from being random. Many factors come in the picture when quantifying a song's success/popularity. This mainly includes the quality of the song, genre, popularity of the artist, theme, and other artistic factors such as the video clip quality and lyrics. This work examines only the correlation between the popularity of a song and its musical features (i.e. musical quality).

There are two main motivations for choosing to work on this problem. First, all our group members are passionate about music, and we thought it would be interesting and fun to apply the concepts we learned in the course on a music related project. The second motivation was realizing the impact and value that the ability of predicting a song's popularity even before its release can add to the music industry: it provides artists a tool to modify/tailor their song composition to generate popular songs (assuming this is their goal), and offer the music advertisement/label companies assistance in investing in a particular artist/song that can generate for them the most success.

To define our project even further, our prediction system will take as input a set of musical features of a song, and will classify the song in one of five groups depending on its expected popularity. We are using the number of views on YouTube as a metric for a song's popularity, and we map these views in five ranges to generate the labels.

Before starting the discussion of our approach to implement the song rating system, we describe in the following section how we modeled our labels and how we

chose the songs, as this is particularly important in our project application. We then report and discuss previous similar work. We also discuss our feature extraction process, the learning methods/algorithms we considered, and finally the learning results.

2. DATA REPRESENTATION AND SELECTION

A. Label Generation

As mentioned earlier, we modeled our learning problem as a classification problem over five classes. And since we are using YouTube views – which can take any integer number up to almost 3 billion – as the success metric, we mapped them to the following ranges to generate the labels:

Views Range ($K=10^3$, $M=10^6$)	Label
0 – 10K	1
10K – 50K	2
50K – 10M	3
10M – 200M	4
200M – 2000M	5

Table 1: mapping between YouTube views ranges and the labels

B. Song Selection

As previously mentioned, out of many factors that can determine the popularity of a song, this project only focuses on modeling the musical (audio) aspect. Thus, properly choosing the songs to be considered for feature extraction is very important, and the following sections discuss the strategies that we followed to mitigate the effect of the other factors as much as possible.

C. Time and The General Music Taste

Over time, music content and composition has changed a lot, and the 'hits' that were very popular in the 60's and 70's for example, are very different than the modern contemporary hit songs. Therefore, it is important, at least for the initial study, to focus on one musical 'era'. Otherwise, the data might have very small correlation with the results. For this reason, we narrowed our choice of songs to after year of 2000.

D. YouTube Popularity

Since we are learning our algorithm based on the number of views on YouTube, another important factor to take into account is the date YouTube started becoming popular. For example, a song could have been very popular right before YouTube was released (and still considered as modern song),

but it did not get enough ‘views’ on YouTube because it has already been out on the radio and people already know the song. The statistics (trends) below show that narrowing down our search to songs released after 2010 (2010 – present) is a good choice.

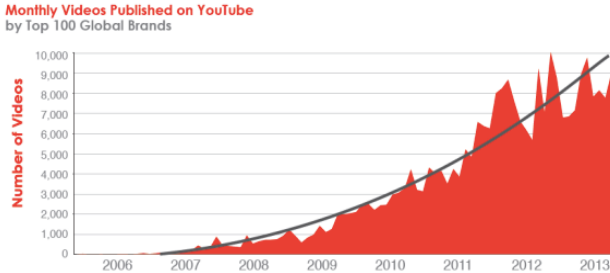


Figure 1: monthly videos Published on YouTube (2006-2013) [1]

E. Genre Diversity

Music comprises a very large number of genres. These genres have different structures and composition processes, which results in (relatively) different song features. Therefore, we also focused on gathering features of songs from genres that are close, in particular pop and alternative rock. The incentive was that this choice might increase the correlation between the input and output data.

F. Artist Style Diversity

Another key difference is the artist’s styles, in particular the composition and musical content of the song. To mitigate the effect of this diversity, we chose a certain number of artists (based on the assumptions above), and we performed feature extraction on a full album instead of individual songs. The idea being that comparing individual songs across a wide variety of artists will give little correlation. In addition, an album has (almost) always relatively popular and unpopular songs for the artist, and this will make it easier to contract this popularity over the difference of the features of these songs.

G. Artist Popularity

It is common that if an artist is already very famous, the user rating will be biased for any new song (even if its quality was not as good as the previous songs). Although this factor is important, it is hard to mitigate. One way to try to reduce its effect is to choose the first album of an artist that became recently popular.

H. Typical Trend of YouTube Views

When choosing a song, it is important to take into account the time and duration since it has been released. This is to make sure that the number of views reaches a relative steady state (or constant growth). According to statistics published by YouTube, this trend is exponential, and reaches a steady state after around 6 months from the release date.

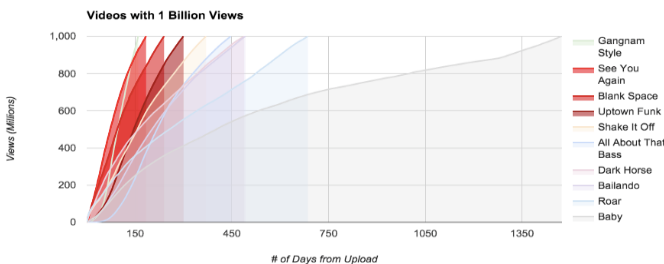


Figure 2: YouTube views versus time (in days) for popular songs [2]

3. RELATED WORK

Li Su, et al. [3] implement a system that classifies guitar playing techniques using features extracted from the audio signal. We looked at this paper hoping to gain some insight into possible features we can extract from the audio content of a song. However, this paper implements a very specialized system and therefore, uses only a limited number of features. We were looking for a broader way of classifying the input songs.

In [4], the author seems to favor metadata of songs such as user’s recommendations over the features extracted from the actual content of the song for music recommendation systems. Therefore, this paper recommends techniques like collaborative filtering which try to determine user preferences from historical usage data. The problem with this approach is that it would be difficult to recommend new songs that do not have much usage data.

We also looked at systems that implemented neural networks for feature extraction. In [5], a deep belief neural network is used to extract features which are then passed on to an SVM for genre classification. The drawback of such a system is that we would need a large number of training examples for such a system to function properly.

In another paper, van den Oord, et. al. [6] implemented a content-based music recommendation system using deep convolutional neural networks. This paper improves upon Collaborative-filtering based approaches to music recommendation (which relies on historical usage data to determine user preferences) by trying to predict the usage data inside one of the layers. This helps overcome the problems faced by Collaborative filtering with new songs for which there is no usage data available. This is a really cool idea. However, the users had access to a very large database of songs (the Spotify song database) and hardware required to implement training of the neural network. Therefore, we decided to take a different approach.

4. DATASET AND FEATURES

In order to extract relevant features, we first needed to decide on both, the type of audio data to process and the features to extract. For the former, we considered three types: MP3, MIDI, and Million Song Dataset (MSD) [7]. We investigated three different types because each had its own advantage and disadvantage as will be discussed next. For the features, we considered both static features as well as time series. However, given the amount of time we spent on data collection and filtering, it was more convenient to work with static features, as working with time series require additional time. We also used a software called jAudio [8] for MP3/MIDI music feature extraction, while the MSD data had the features already extracted.

A. MP3 and jAudio

After a thorough search for audio features extraction software, we chose jAudio for its ability to extract a large set of audio features and its compatibility with both MP3 and MIDI songs. jAudio is part of a digital signal-processing project. It was developed to extract several audio properties of songs (e.g. beat points and statistical summaries) tailored to machine learning algorithms that aim to predict the ‘hit’ degree of songs. As for the features, jAudio provides static

features such as the FFT of the song, strongest beat, harmonic spectral centroid, and others. It exports the extracted features into excel sheets (type double and integer). Also, before passing this data to our learning algorithms, we normalized it (divided by the maximum value of the corresponding range) especially for magnitude-dependent learning algorithms.

A main advantage of MP3 songs over MIDI and MSD is that it contains all the information about the song: instruments, pitch, vocals, and others. However, during MP3 data collection, we faced several difficulties. First, even though we wrote a script to automate the download, we needed to download many other songs individually from the Internet. Another problem was that jAudio was able to process only a portion of the MP3 songs (around 20%), and we had to manually input the songs one by one for feature extraction. The previous two issues left us with around 242 songs with extracted features (out of almost 1200). Given that jAudio provides up to 260 features, we wished to have more songs to avoid over-fitting in case we needed to use a large number of features for higher accuracy. Another main issue was noise. It was difficult to determine how much noise an MP3 song contained (glitches, etc...). Even after passing the songs through filtering programs, it was still hard to determine how much the noise can affect the results.

B. MSD

Another dataset we considered is the Million Song Dataset (MSD). MSD is a set of audio features and metadata for one million songs provided by the Echo Nest. It offers a great advantage over MP3 and MIDI songs in terms of the size of the dataset (1 million) and the features that are already extracted. In total, it offers 24 features that we can choose from. The only main disadvantage is that it does not provide flexibility in extracting new features for the songs, and does not give prior knowledge of the noise. Among the features MSD provides, we name the key, mode (major/minor), mode confidence, genre, year, and others. Most of the features can be considered as static features.

The MSD dataset is very large (~280 GB). Luckily the developers provides a subset of 10 000 songs that we could download within reasonable time. Interfacing MSD dataset with our algorithms required developing python, Perl, and MATLAB scripts. We also used them to filter out the songs released before 2010 and other criteria (discussed earlier in the song selection section). After filtering, we ended up with 1667 songs.

C. MIDI

Even though MIDI songs have no noise, they do not contain the vocal track, which could be important for a song's success. We were also not able to obtain a large dataset online, and thus we decided to work with the two previous options.

D. Label Extraction

To get the number of YouTube views, we developed a script that takes a list of songs, fetches the corresponding YouTube webpage and extracts the number of views for the first song in the YouTube list, which has usually the highest

number of views. The script then normalizes the numbers in the corresponding ranges and labels the data (1 – 5).

5. DATA EXPLORATION

After extracting the features, and before running our learning algorithms, we worked on visualizing our data by performing principle component analysis (PCA), and calculating the mutual information for the features.

PCA helped us observe how the data is clustered, whether it is linearly separable, and based on the result determine the type of kernel we need to use in our learning algorithm. To know the size of the basis to project our data on, we plotted the variance captured by the different principal components and found that most of the information (variance) is in the first 3 – 4 components for MSD and MP3 features (shown below is MP3 case). Therefore, having a 3D figure should give us a good representation of the total variance. The PCA plots which we show below for different classes did not show a very clear clustering of the data, suggesting that we need to use a kernel different than the linear one (confirms with the results we discuss in the next sections).

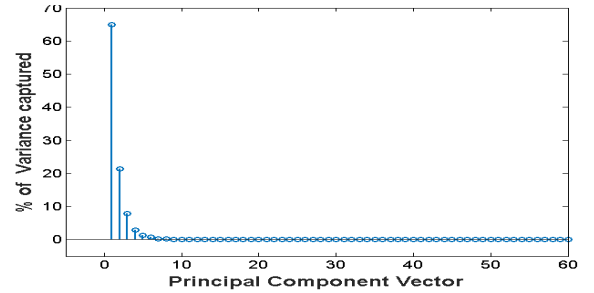


Figure 3: percentage variance captured by the PC vectors (MP3)

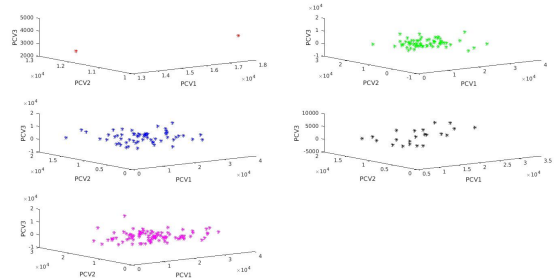


Figure 4: data for different classes projected on 3-dim using PCA (MP3)

In addition to PCA, we calculated the mutual information for the different features. This helped us rank them for the cross validation experiments (discussed in the results sections). The plots below show the mutual information score (after normalization) versus the feature index for jAudio (MP3) and MSD datasets.

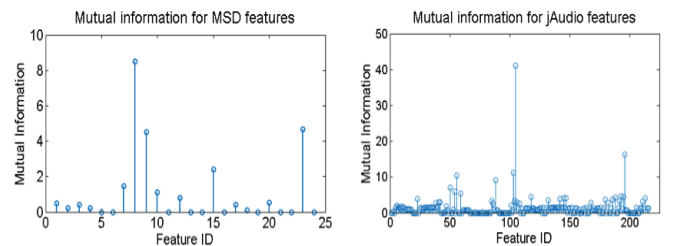


Figure 5: Mutual information for MSD and jAudio features

6. METHODS

In our project, we used three different learning algorithms to train our classifiers: Gaussian Discriminant Analysis (GDA), modified GDA, and Support Vector Machines.

A. Gaussian Discriminant Analysis

Given that most of our features are real valued and extracted from the audio signal of the song, we made a fairly reasonable assumption that the feature vector has a multivariate normal distribution. GDA is known to be asymptotically efficient if the feature vector is Gaussian and this was our main source of motivation to proceed with GDA.

Gaussian Discriminant Analysis models the distribution of the data points (the $x^{(i)}$'s) given information about their corresponding class labels (the $y^{(i)}$'s) as being multivariate Gaussian. For our project, we implemented a five-class classifier (that is $1 \leq y^{(i)} \leq 5$) and used two variations of GDA. In the first variation, we assumed the covariance matrix, Σ to be the same for all classes. Therefore, the conditional of $x^{(i)}$ given $y^{(i)}$ is given by,

$$p(x^{(i)}|y^{(i)} = j) = \frac{1}{(2\pi)^{\frac{n}{2}}|\Sigma|^{\frac{1}{2}}} \exp \left\{ -\frac{1}{2} (x^{(i)} - \mu_j)^T \Sigma^{-1} (x^{(i)} - \mu_j) \right\}$$

where 'n' is the dimension of the feature vector (so each $x^{(i)} \in \mathbb{R}^n$), μ_j is the mean of each class and Σ is the covariance matrix. Using Bayes' rule, the conditional distribution of $y^{(i)}$ given $x^{(i)}$ is then found to be,

$$p(y^{(i)}|x^{(i)}) = \frac{p(y^{(i)})p(x^{(i)}|y^{(i)})}{\sum_j p(y^{(i)} = j)p(x^{(i)}|y^{(i)} = j)}$$

Given a new song's feature vector x , a prediction is made by finding,

$$y_{predicted} = \arg \max_{1 \leq j \leq 5} p(y = j|x)$$

The parameters of this model are then the class prior probabilities $\phi_j = p(y^{(i)} = j)$, ($1 \leq j \leq 5$), μ_j and Σ . The optimal estimate of these parameters that maximizes the log-likelihood of the data can now be obtained as follows,

$$\begin{aligned} \phi_j &= \frac{1\{y^{(i)} = j\}}{m}, 1 \leq j \leq 5 \\ \mu_j &= \frac{\sum_{i=1}^m 1\{y^{(i)} = j\} \cdot x^{(i)}}{1\{y^{(i)} = j\}}, 1 \leq j \leq 5 \\ \Sigma &= \frac{\sum_{i=1}^m (x^{(i)} - \mu_{y^{(i)}})(x^{(i)} - \mu_{y^{(i)}})^T}{m} \end{aligned}$$

B. Modified Gaussian Discriminant Analysis

In our second variation of GDA, we used different covariance matrices for each class. Therefore, all parameters would be updated in the same way as before except Σ , which would now be updated separately for each class as follows,

$$\Sigma_j = \frac{\sum_{i=1}^m (x^{(i)} - \mu_{y^{(i)}})(x^{(i)} - \mu_{y^{(i)}})^T}{1\{y^{(i)} = j\}}, 1 \leq j \leq 5$$

C. Support Vector Machines

The Support Vector Machine learning algorithm is inherently a binary classifier that tries to find a hyperplane that separates data points belonging to two classes. This

hyper-plane is found by solving the following optimization problem,

$$\begin{aligned} \min_{\gamma, w, b} & \frac{1}{2} \|w\|^2 \\ \text{s. t. } & y^{(i)}(w^T x^{(i)} + b) \geq 1, i = 1, 2, \dots, m \end{aligned}$$

To use the SVM for our five-class classification problem, a one-vs-one implementation is used [9]. In this implementation, a new binary classifier (which in this case is an SVM) is learned for each pair of classes. This means there are $5(5-1)/2 = 10$ SVM classifiers which are learned. During prediction, each classifier (or binary learner) makes a prediction and the class which gets the largest number of predictions (or votes) is output as the final predicted class.

7. EXPERIMENTS AND RESULTS

In this section, we discuss some experiments we ran in order to make some informed decisions about the choice of hyper-parameter values for our learning algorithms such as the optimum number of features, the type of Kernel function to use and so on. Using these tuned parameter values we then discuss the results obtained with the three learning algorithms.

A. Experiment 1: K-Fold Cross-Validation

All parameters of the GDA learning algorithm (both with the same and different class co-variances) are learned during training. However, there are some things that cannot be explicitly learned such as the number of features to use. Therefore, we used filter-feature selection with K-fold cross validation ($k=10$) to help us choose the optimum number of features. In order to do so, we varied the number of features from one to the maximum number of features available for each data set (216 for MP33 and 24 for MSD). For a particular number of features chosen (say 'N'), we found the features with the 'N' best mutual information scores. We then found the optimal number of features as the one that gives us the best average accuracy using k-fold cross-validation. We ran this procedure with the Million Song Dataset and the MP3 song dataset separately.

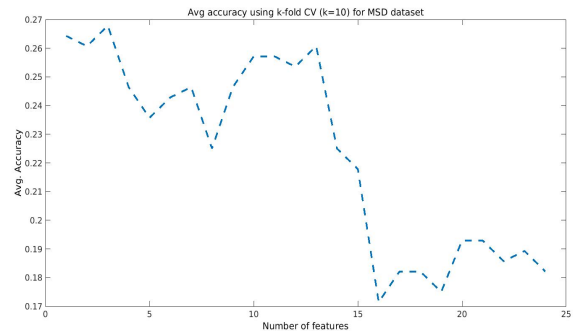


Figure 6: k-fold cross validation for MSD features ($k=10$) starting with the highest mutual information scores

Figure 6 shows a plot of the average accuracy versus the number of features chosen for the MSD dataset obtained using the GDA classifier. As can be seen in this plot, beyond a point the accuracy falls sharply as the number of features increases. This may be due to redundant features being added which would increase the risk of over-fitting. The optimal number of features is found to be 3 for the GDA classifier. For the MP3 dataset, however, the k-fold cross-validation accuracy seemed to be increasing as the number of features was increased. However, using more than 50 features creates

a singular covariance matrix, which results in erratic results and a very low accuracy when a larger test set is used. Therefore, we chose the optimum number of features for the MP3 dataset to be 50 for the GDA classifier. The trend would be similar for the modified GDA classifier with different class co-variances and so would the corresponding optimum number of features.

B. Experiment 2: hold-out Cross-Validation

For the SVM classifier, we used all the features available in each dataset. When we ran the SVM with a linear kernel, we were getting a lower accuracy than the GDA and GDA modified classifiers. Rather than reduce the number of features, we decided to try using a Gaussian Kernel. Our motivation for doing so was that we had obtained a higher accuracy with GDA which assumes that the distribution of the feature vector is Gaussian. Furthermore, during initial runs of the SVM using linear and Gaussian Kernels, the run time was significantly lower with the Gaussian Kernel as compared with the linear Kernel. This suggested that the optimization problem for SVMs was easier to solve using a Gaussian Kernel and this was our main source of motivation for choosing the Gaussian Kernel. This also confirms with the PCA results we obtained before. We also decided to specify a cost-matrix, C for misclassification where

$$C(I, J) = \text{Cost}(\text{Predicting } x^{(i)} \in \text{class } I \text{ to be in class } J) \\ = 1000, \quad \text{if } I \neq J$$

and $C(I, J) = 0$, if $I = J$.

In order to test the performance of these different approaches, we used holdout cross-validation where we trained on 70% of the data set (randomly chosen points) and tested on the remaining 30% of the data. We ran the above procedure and obtained the classification accuracy over the test data for three cases:

- SVM classifier with a linear Kernel.
- SVM classifier with a Gaussian Kernel.
- SVM classifier with a Gaussian Kernel and the cost matrix C specified.

The classification accuracies obtained for each of these classifiers on the MSD dataset is shown in figure 7. From the figure, we can see that the best accuracy was obtained using the SVM with a Gaussian Kernel and the Cost Matrix C .

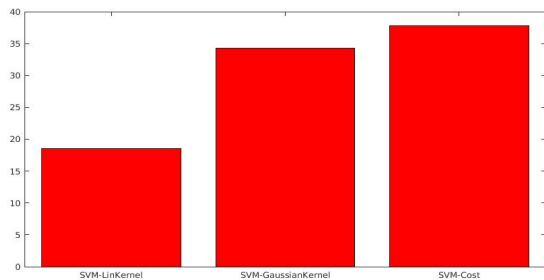


Figure 7: accuracy with different SVM hyper-parameters

C. Results

Using the tuned values of these parameters, we ran a simulation to compute the classification accuracy of each classifier using the MSD and MP3 datasets. We used classification accuracy to be the main metric to judge the performance of each classifier. Each classifier was trained on 70% of the data set (randomly chosen points) and tested on the remaining 30% of the data. In order to make a fair comparison, all the classifiers were trained and later tested on

the same data points for each data set. The results we obtained are shown in Table 2. For both the MSD and the MP3 data sets, the SVM obtains the highest classification accuracy. For the MSD data set the performance of GDA and the Modified GDA classifiers are similar with GDA doing slightly better. However, for the MP3 dataset, the Modified GDA classifier does significantly worse. The reason for this may be the following. The sample size for this data set is fairly small. Therefore, the training set may have been imbalanced meaning the co-variance matrices for the minority classes are not accurate. Hence, if these classes appeared in the test set, they will most likely be misclassified.

Apart from classification accuracy, another method we used to compare the performance of the different classifiers was to see how much far the erroneous predictions were from the actual predictions. We obtained this by breaking down the number of misclassified test data points into percentages of points that were only misclassified by one class, by two classes and so on.

The results obtained can be seen in Table 3. From Table 3, we can see that for the MSD data set, all the classifiers misclassify by one class majority of the time (~66-71%). As our system only gives a rating for a song, a misclassification by one class is not a big issue. For the MP3 data set, we see that the GDA with different class co-variances misclassifies by one class majority of the time (~36%). However, the SVM (although more accurate) misclassifies by two classes most of the time (~49%).

Classifier	MSD	MP3
GDA	33.52 %	29.0278 %
Modified GDA	32.8 %	19.3056 %
SVM	37.8 %	40.28 %

Table 2: Classification accuracy

Diff between true & pred. class	MSD Dataset			Mp3		
	GDA (%)	Mod. GDA (%)	SVM (%)	GDA (%)	Mod. GDA (%)	SVM (%)
1 class	68.92	66.76	71.38	35.62	53.53	25.59
2 class	17.93	18.3	16.08	34.63	32.01	48.84
3 class	12.91	13.81	11.9	28.57	14.29	23.26
4 class	0.24	1.13	0.64	1.17	0.17	2.31

Table 3: Misclassification Error

8. CONCLUSION

In this work, we implement a five-class song popularity classifier based on well-defined musical features. SVM algorithm showed to give a better performance compared to GDA (in its two versions) mainly because it has a lower bias in general than GDA. Our future work will involve obtaining more MP3 data to learn from, and investigating other features including time-series, which could give even higher accuracies.

REFERENCES

- [1] Study: The Top 100 Global Brands Winning On YouTube Publish 78 To 500 Videos A Month. (2013, August 7). Retrieved December 12, 2015,

- from <http://marketingland.com/study-top-100-global-brands-winning-on-youtube-publish-anywhere-from-78-to-500-videos-a-month-54934>
- [2] Billion-view videos are happening faster on YouTube. (n.d.). Retrieved December 12, 2015, from <http://youtube-trends.blogspot.com/2015/10/billion-view-videos-are-happening.html>
- [3] L. Su, L.-F. Yu, and Y.-H. Yang. Sparse cepstral and phase codes for guitar playing technique classification. ISMIR, 2014[2] - Slaney, M., "Web-Scale Multimedia Analysis: Does Content Matter?," in *MultiMedia, IEEE*, vol.18, no.2, pp.12-15, Feb. 2011
- [4] Slaney, M., "Web-Scale Multimedia Analysis: Does Content Matter?," in *MultiMedia, IEEE*, vol.18, no.2, pp.12-15, Feb. 2011
- [5] (n.d.). Retrieved December 12, 2015, from <http://ismir2010.ismir.net/proceedings/ismir2010-58.pdf>
- [6] Van den Oord, Aaron and Dieleman, Sander and Schrauwen, Benjamin, "Deep content-based music recommendation", *Advances in Neural Information Processing Systems* 26, 2643--2651, 2013, Curran Associates, Inc.
- [7] Welcome! (n.d.). Retrieved December 12, 2015, from <http://labrosa.ee.columbia.edu/millionsong/>
- [8] JAudio. (n.d.). Retrieved December 12, 2015, from <http://jaudio.sourceforge.net>
- [9] 1.12. Multiclass and multilabel algorithms¶. (n.d.). Retrieved December 12, 2015, from <http://scikit-learn.org/stable/modules/multiclass.html>