Learning How to Pick Singles for Pop Stardom

Ben Holtz BHOLTZ@STANFORD.EDU
Stanford University, Palo Alto, CA 94306 USA

1. Introduction

1.1. Pop Singles

What do Stevie Wonders' "Isn't She Lovely?", Jimi Hendrix's "Voodoo Chile" (not to be confused with "Voodoo Chile (Slight Return)"), and Sufjan Stevens' "Casimir Pulaski Day" all have in common? While, your first response should be, correctly, that they are incredibly good songs, it might surprise you to know that none of these classics were released as singles. Surely Stevie's producers heard "Isn't She Lovely?" - did they really think it would be less popular than "As"? Of course, studios usually make good choices as to which songs should be singled out, but why not automate this process, hopefully identifying along the way components of songs that are crucial to popularity and success.

Many others have tried to automate musical tasks, like evaluating track similarity (Berenzweig et al., 2004), genre classification (Tzanetakis & Cook, 2002), or hit detection (Dhanaraj & Logan, 2005), but no work has addressed the problem of putting an album's best foot (or feet, in case of multiple singles) forward. Although others have addressed different problems, we use similar methods, namely representing music by harmonic, rhythmic, and lyrical features.

2. Data Acquisition

2.1. Lyrical features

While for now I am using lyrics copied from any of the myriad lyrics websites, I also have access to the Million Song Dataset (Bertin-Mahieux et al., 2011), a collection of lyrics data already prepared for use in learning tasks (it is in bag-of-word format, stemmed, and available in a SQLite format). I also have access to the MusixMatch API, which gives free, albeit limited, access to the MusixMatch database of lyrics.

Each song's lyrics are treated as a bag of words, and then the following features are extracted:

1. Averge word length. I use the 2 norm to increase the weight of longer words.
2. Number of unique words. The total number of unique words used in the lyrics. Songs with more unique words are generally more complex, think Paul Simon ('The Mississippi delta was shining like the national guitar...') versus Justin Bieber ('Baby, baby, baby, oh...').
3. Number of total words. This gives us some idea of the relative number of lyrics in the song and therefore the length of the vocals in the song.

2.2. Audio features

Audio features are not yet in use, but will be extracted using the OpenSMILE audio feature extraction toolkit (Eyben et al., 2010). See 5.2 for more on what type of features can be extracted.

3. Methods

3.1. Single albums

It is relatively easy to learn a model that correctly predicts the singles in single album. Albums are generally small, and usually have only one or two singles, so often times a quadratic model, like SVM with a quadratic kernel, will be able to fit a decision boundary with no learning error.

3.2. Multiple albums

Of course, every album is different, so we want to find the most general model of all of the models learned on each of the single albums. It wouldn't do to compare models fit on very different albums, say Kanye West's *Yeezus* and Daft Punk's *Discovery*, so we need to normalize the models. For each training instance $x^{(i)}$, we replace its jth feature $x_j^{(i)}$ with

$$\frac{x_j^{(i)} - \mu_j}{\mathrm{Var}(x_j)}$$

Next, we want our model to learn how to incorporate each of the models from the different albums into one model. We do this in two ways. Our first method is AdaBoost, using the SAMME algorithm as described by (Zhu et al., 2006) and as implemented in `scikit-learn` (Pedregosa et al., 2011).

The AdaBoost algorithm works roughly as follows:

1. Each of the n observations has its weight initialized to $\frac{1}{n}$
2. Fit a classifier from a family of weak classifiers to the observations
3. Compute the weighted error of the classifier
4. Reweight the observations, giving missclassified observations higher weight
5. Repeat the past two steps for as many classifiers in the family as you like
6. Classify an instance into class k so that the sum of the errors of all the weak classifiers is minimized

We can apply this method to the set of all songs in the training set, after they are renormalized in feature space.

The second way we incorporate multiple albums is by chaining together k-nearest neighbors and SVM as follows:

1. Given an album A, find the average $\hat{a}$ of its songs in feature space. This is the representative "song" of the album.
2. Find the k albums in the training set whose representative in the feature space are closest to $\hat{a}$
3. Allow each model fit by the nearest neighbor albums to "vote" on the normalized songs of A, so that the prediction for each song is the majority vote of the k classifiers.

Note that we use SVM with the radial basis function kernel, allowing us to caputer the one or two out of many.

4. Results

We use only a small dataset, 3 albums: Stevie Wonder's *Songs in the Key of Life* and *Talking Book* and Weezer's *Weezer* (better known as the Blue Album).

5. Future work

5.1. More data

Data was surprisingly difficult to come by. Even the large, freely available data sets with lyrics are incomplete, and work still has to be done by hand to label songs as singles. In the future, it would be helpful to scrape a site like `www.azlyrics.com` to get the lyrics I need.

5.2. Audio Extraction

I have OpenSMILE working, but it is difficult to specify which type of feature generation to use. Also, this is inherently a challenge because of the huge variety of features that can be generated. For example, OpenSMILE can extract, from a single song, signal energy (over all intervals of the song), loudness, voicing probability (how much of the signal is human voice), etc.

References

Berenzweig, Adam, Logan, Beth, Ellis, Daniel PW, and Whitman, Brian. A large-scale evaluation of acoustic and subjective music-similarity measures. *Computer Music Journal*, 28(2):63–76, 2004.

Bertin-Mahieux, Thierry, Ellis, Daniel P.W., Whitman, Brian, and Lamere, Paul. The million song dataset. In *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011)*, 2011.

Dhanaraj, Ruth and Logan, Beth. Automatic prediction of hit songs. In *ISMIR*, pp. 488–491, 2005.

Eyben, Florian, Wöllmer, Martin, and Schuller, Björn. Opensmile: the munich versatile and fast open-source audio feature extractor. In *Proceedings of the international conference on Multimedia*, pp. 1459–1462. ACM, 2010.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

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

Zhu, Ji, Rosset, Saharon, Zou, Hui, and Hastie, Trevor. Multi-class adaboost. *Ann Arbor*, 1001 (48109):1612, 2006.