An Evaluation of Artist Recognition Methods

Gabriel Zalles, Dror Ayalon

***Abstract*—Artist recognition aims at recognizing the artist or artists responsible with the creation of a particular piece of music from feature vectors extracted from audio signals. Unlike other algorithms which pair data with metadata, providing users with more than just information about the artist, recognition tasks assume that the data to be categorized cannot be found in the larger set and, instead, attempt to make an accurate prediction regarding the artist responsible for a piece of music based on former works by the artist. This task has been compared to speaker recognition and instrument recognition partly because of the success researchers have found when implementing similar techniques across tasks. This paper evaluates the accuracy of an artist classification system using numerous timbral feature extraction methods, post-processing techniques and classification algorithms. All of the code, as well as the database, are freely available online.**

I. Introduction

A. Background

Before recapitulating the methodology and findings of this paper we should discuss the motivation of artist classification systems as well as any problems surrounding the task. The most obvious use of a system like this is for the labeling of copyright infringement. A system such as this can be used to flag samples, extracted from other songs, cover songs or remixes that seek to produce revenue from music produced by other artists without awarding royalties. [21] provides a rich description of the most common examples, including these, and many other scenarios. Another interesting usage for a system like this is in the field of musicology. Using algorithms like this one we can study musical trends and development of genres over large populations [9]. This system can also be used for data management. The average iTunes collection in 2006 was 3,500 songs [2]. Today, many people rely on subsamples of even larger databases provided by services like Spotify, Tidal or Pandora. Companies such as these are investing to ensure that copyright infringement is caught, their databases are up-to-date, and no duplicates of songs are posted by unauthorized accounts.

B. Problems with Artist Recognition

With the regard to some of the inherent problems in artist recognition. The first and most salient is that of defining who the artist of a song is. While it is often the case that the the artist attributed to a song is the singer performing the lyrics of the song, this is not always the case. Today, we have the pleasure of being able to enjoy music from a large number of artists who are not singers at all. Some artists are instrumentalist, others are multi-instrumentalists. Some are bands while others strictly use machines and algorithmic processes to create music. Consider this real world example: a recent release by an electronic music producer featured over a dozen singers in his album yet the producer himself, the main artist attributed for the work, sung in none of those songs. The entire album is his, yet every song features an artist whom has his or her own catalog of material which could theorically be included in dataset. While cases such as this might seem more like an exception than the rule they are important to point out as they give insight into just how monumental the task of creating an accurate artist recognition system is.

C. Literature Review

In [1], the author showed that there was an increase in performance, during the artist recognition task when segmentation was done prior to artist classification in vocal centric songs. Segmentation allowed researchers in this paper to split data into instrumental sections and vocal sections. While these results are promising, Berenzweig also showed that performance suffered by 35% when training and tests sets were selected from multiple albums. This suggests that these systems learn about album-specific properties. The segmentation method would also be problematic in cases when instrumentalist change the tone of their instrument over several songs, albums, or in cases when a multi-instrumentalists plays different instruments in a single song. Overall, and intuitively, vocal parts seem to possess a rich spectral signature that makes them inherently apt for artist recognition task: a feature exploited by Berenzweig. For other genres, such as jazz, listeners rely on motifs and stylistic features to identify performers. In other genres, such as in classical music, a dynamic and temporal fluctuations, native to a composer, can be used to extrapolate an accurate prediction. This suggests that, in order to apply the right classification method, whether that be temporal or spectral, one must first know the general genre in which one is operating and distinguish the most salient attributes used by those who enjoy those genres to recognize an artist. In the future, we should also be weary of algorithmic systems which aim to exploit these stylistic features to imitate popular music such as David Copes EMI [7] which has been found to fool both categorizing systems and expert listeners.

A 1999 paper by Brown [4], on the recognition of musical instruments, came to similar conclusion. In this paper, Brown points out that when subjects were asked to classify recordings as either produced by an oboe or a saxophone, subjects reported that the style of the pieces greatly affected their responses. Pieces that were in the style of jazz were labeled as having come from a saxophone and classical sounding pieces were labeled as oboe recordings. While their system did just as well or better than humans, it is important to note since many proposed systems could benefit from a macro-level temporal model which clusters numerous predicted features of songs in order to make better observations.

Another key piece of literature in our paper, due to the use of the database *artist20*, is that of Ellis from Columbia University [11] . While working at LabROSA, Dan and his colleagues curated a dataset specifically for the purpose of artist recognition. The database is a subset of the larger *uspop2002* dataset which was expanded to cover the needs of their study. The database consists of of 1,413 songs from twenty artists. Each artist is represented by a total of six albums with an average of ten songs. In their study, disjoint albums were selected for training and testing phases to avoid features related to mastering: our aformentioned album-specific features. The authors also selected albums which reflected the least amount of change in style across albums. While the set contains mainly rock albums, it should be noted there is large gamut of styles represented due to the wide range of expressions all belonging to the genre. A number of partitioning methods are also proposed by the authors in the curation of training and testing cuts. The first is a 3-2-1 method in which three albums are used for training, two for testing and one for validation. The second is a 6-fold jacknife scheme in which, for each artist, 5 albums are used to train the model and one is used for testing/verification. The album used for testing is the rotated from the pool of albums and the average accuracy across all six trials and all 20 artists is reported in a confusion matrix. In our testing we relied on a 80/20 split for the training and classification of data. The model was trained used 80 randomly selected musical track from the 1,413 song database while 20 percent of the files were used for testing/validation.

II. Theory

A. Feature Extraction

1) MFCCs: A number of feature extraction methods exist for the classification of musical artists. Typically, timbral features are selected in lieu of temporal features. The most common timbral features exploited in the literature are FFT coefficients, cepstral coefficients, mel-frequency cepstral coefficients, zero-crossing rate, sprectral spread, spectral centroid, spectral rolloff and LPC (linear predictive coding) coefficients. Wavelet analysis was also examined by Lambrou et al. in 1998 [16]. That being said, it seems, from a preliminary survey, that the most common feature used in the detection of artist is the Mel-Frequency Cepstrum Coefficient (MFCC). MFCCs are short-time spectral decompositions of an audio signal which convey the general frequency characteristics important to human hearing [18]. They are acquired by mapping audio signal to the mel scale, a perceptual pitch scale developed in the 1940s by Stevens [22]. It is used, much like many other methods in machine learning today, as a biomimetic tool determined to describe, for machines, the way humans perceive the world. Mathematically mel frequencies can be described as:

$$mel = 1127.01028 * ln(1 + f/700)$$

The formula was derived by asking subjects to evaluate the relative frequency of two pitches using 1000 Hz as the reference. Any tone perceived twice as high is demarked as 2000 mels and any twice as low as 500 mels. The relationship between mel and Hz can be seen in Fig. 1. Using the mel scale we can create a set of triangular filterbanks to which our audio signals are mapped. The results give us the average energy at each filterbank and is calculated via a matrix multiplication between the output of an STFT and the filerbank. After this process, the cepstrum is acquired. The cepstrum corresponds to the *fourier transform* of the log magnitude spectrum (dB) [19] and is acquired by applying a Discrete Cosine Transform (DCT) unto the mel FFT coefficients. Mathematically the DCT can be described as:

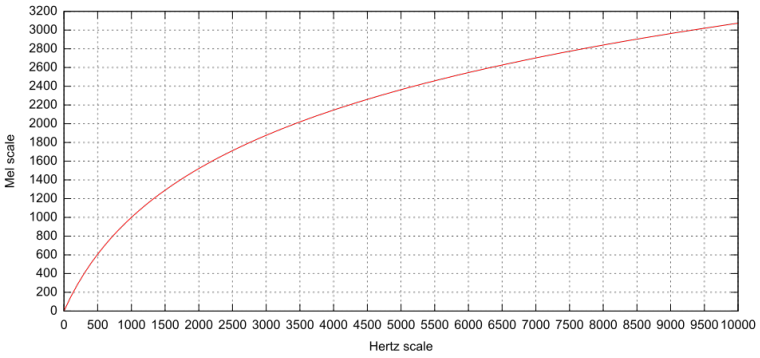

Fig. 1. Mel v. Hz plot

$$X_k = \sum_{n=0}^{N-1} x_n \cos\left[\frac{\pi}{N}\left(n + \frac{1}{2}\right)k\right] \qquad k = 0, \ldots, N-1.$$

The output of the DCT gives us a number of decorrelated coefficients equivalent to the number of filters used to aggregate the energy at each mel band. Generally, these mel filters overlap, much the same way our critical bands do, and are normalized to unity sum. The word cepstrum corresponds to an anagram originating from the word spectrum. In the literature the anagrams *quefrency* and *liftering* are also used. MFCCs have been successfully used in the past for the task of speaker identification [20]. Work by Logan [17] has also shown the robustness of MFCCs in artist classification.

2) Spectral Contrast: In order to try and improve our results, we decided to also apply the Spectral Contrast feature extraction method to our selected database. Spectral Contrast (SC) was proposed by Jiang et al. [14] and represents the spectral characteristics of a music clip. Jiang showed improved results in classifying audio clips using Spectral Constrast when compared to MFCCs. The Spectral Contrast uses an octave-based approach to characterize audio segment (in our case, 7 octaves classes were used). Therefore, applying a DCT process, with identical transform length to the MFCC, on the Spectral Contrast feature matrix, allows us to stack these features together to a single matrix. This process gives us the option of using both MFCC and Spectral Contrast features as input data for our classifiers. Spectral contrast features roughly reflect the relative distribution of harmonic and non-harmonic components in the spectrum. In contrast, MFCCs give an average the spectral distribution in each sub-band, which result in a loss of relative spectral information when compared to SC. Relative spectral changes are preserved by

using a peak/valley selection method. The strenght of spectral peaks and valleys are estimated by the average value in a small neighborhood around maximum and minimum values respectively instead of exact max and min values themselves. In lieu of a DCT transform the Karhunen-Loeve transform is used in this process. The K-L transform is closely related to PCA and widely used in data analysis. The basis vector if a K-L tranform are the eigenvectors of the feature's covariance matrix. Using KLT we can diagonalize the covariance matrix, removing the correlation of neighboring features [8]. A depiction mel filterbanks, with octaves separation on the x-axis can be seen in figure 2; octave-based filterbanks would be symmetrical if represented in this figure.

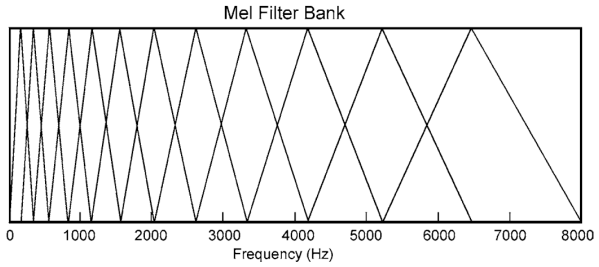

Fig. 2. Mel FB with octaves depicted on x-axis

B. Post-processing

One common step in the creation of machine learning algorithms such as this is the post-processing of data. Post-processing, as the name suggests, is the process of processing the data after feature extraction in some way with the aim of improving classification results or increasing the speed of the classifying algorithm by reducing the dimensionality of the data. Much like in the feature extraction step, there are a myriad of post-processing algorithms which are used to improve performance, KLT being just one of them. In [5] Bruha and Famili present the importance of post-processing in Knowledge Discovery of Databases revealing that, while post-processing might seem like a trivial stage in the artist recognition task, it is imporant to understand and justify any post-processing methods used one's machine learning algorithms. One of the most commmon techniques is the smoothing of the data via a low pass filters. Peak picking, using a variable threshold, is also used to find onsets in musical signals. For the post-processing of the MFCC feature vectors in our experiment a Principal Component Analysis process was adopted. PCA is a linear dimensionality reduction technique, similar to KLT, which uses Singular Value Decomposition (SVD) to project data to a lower dimensional spaces [25]. By using PCA we can find the linear combination of features with the covariance. In other words, we can use PCA to select which features will gives us the most information and thus the highest likelihood of accurate classification.

C. Classifiers

1) SVM: Support Vector Machines, or SVM for short, are one of a myriad of classifying algorithsm used in MIR for the identification of artists. An SVM maps data into a higher dimensional input space and constructs an optimal separating hyperplane in this space [23]. SVMs is a supervised learning method, this means that training data is involved in the process of developing the model for classification. In contrast to unsupervised learning, where the system finds relationships between data itself, supervised learning models depend on labeled data. One of the disadvantages of SVMs is the need for cross-validation in order to arrive at probability estimates [6]. Our classifying algorithm made use of SVM and compared it with results from using the random forrest method. Several types of kernels were selected to exae their ability to recognize artists succesfully from the database.

2) Random Forest: Much like Jin and Bie [15], our system attempted to evaluate the suitability of PCA and random forrest classification methods in a discriation task. Jin and Bie used these techniques to a similar task, that of genre classification. As they explain, random forrest, developed by Leo Breiman [3], uses an ensemble of classification trees built using a bootstrap sample of the data. Bootsrap is a computer-based method for assigning measures of accuracy to sample estimates [10]. This technique allows estimation of the sample distribution of almost any statistic using only very simple methods [26]. Random forests are a way of averaging multiple deep decision trees, trained on different parts of the same training set with the goal of reducing variance [12]. This comes at the expense of a small increase in the bias and some loss of interpretability, but generally greatly boosts the performance in the final model.

3) AdaBoost: An AdaBoost classifier is a meta-estimator that begins by fitting a classifier on the original dataset and then fits additional copies of the classifier on the same dataset but where the weights of incorrectly classified instances are adjusted such that subsequent classifiers focus more on difficult cases. The scikit-learn library that we used make use of a multi-class AdaBoost process developed by Zhu et al. [13].

III. Implementation

For our implementation we used Python and a number of free and openly available online libraries that would allow us to: manipulate the data more easily, perform numerical operations, plot the data, and run our model. The feature extraction was performed manually for MFCC and compared during classification against the library implementation of the feature extraction method. In order to get our own MFCCs, filterbanks were manually created. A set of linearly spaced values between the minimum and maximum frequency were computed and mapped to the mel scale. The minimum and maximum values selected for our mel filterbanks was informed by the segmentation procedure described in the literature review. The bandwidth selected represents what we believe is the typical range of frequencies generally found in vocal performances across musical pieces. Thus, our intention with the selected range is to isolate the vocal frequency range extracting the rich spectral features of the voice endemic in artist classifying systems. The parameters values used for the creation of our MFCC feature vectors can be seen on table I.

TABLE I
MFCC PARAMATER VALUES

Window Size	2048
Hop Size	256
Minimum Frequency	500
Max Frequency	5000
Number of Mel Filters	40
Number of DCT Coefficients	15

1) Code Structure: All of the code pertaining to the implementation of this classification model was written in Python3. Our application mainly consisted of the following three files:

- `data_manager.py`: A Python class that contains helper functions and utilities for: reading the classes, reading the audio files, analyzing audio files (applying MFCC and Spectral Contrast feature extraction methods), calling PCA transformation on the audio data, and running the classification.
- `classification.py`: A Python script the contains a complete procedure of classification, with a variety of properties, and production of a report in CSV file format with our classification results.
- `main.py`: A Python script that contains the order of operation of our application. On this script, we call the different functions from `data_manager.py` and `classification.py`.

We also wrote two functions, `get_class_names()` and `get_track_paths()`, to parse the data and the paths to the files we got from the artist20 dataset. These are used to navigate to the list files provided in the dataset and parse the strings in order to iteratively read every 32kpbs, mp3 file.

2) Spectral Feature Extraction Functions:

- `compute_mfccs (filepath, win_size, hop_size, min_freq, max_freq, num_mel_filts, n_dct)`: This function returns an array of MFCC matrices for a given file list, based on the given parameters. This MFCC algorithm was written and implemented by us.
- `mfcc_librosa(file_list)`: This function returns an array of MFCC matrices for a given file list, based on hard-coded parameters. This MFCC algorithm uses librosa (MIR Python Package) implementation of the MFCC algorithm. This function was used to compare and monitor our MFCC algorithm with an algorithm that is being used widely in the industry. We used this function to compute a 40 filter MFCC matrix.
- `spectral_contrast(file_list)`: This function returns an array of Spectral Contrast matrices for a given file list and use the parameter values `(win_size=2048, hop_size=512)`. This function uses librosa's implementation of the Spectral Contrast algorithm.
- `mfcc_spectral_contrast(file_list)`: Given a file list, this function returns an array of matrices that contains MFCC data with `(win_size=2048, hop_size=512)` and Spectral Contrast data `(win_size=2048, hop_size=512)` stacked together. This function allowed us to check if combining the Spectral Contrast and the MFCC to a singular feature matrix could produce better classification results.

Data post-processing functions `pca(array)`: This function uses Scikit-learn's PCA class to compute PCA on a given array of data. This function uses the LAPACK implementation of the full SVD or a randomized truncated SVD by the method of Halko et al. 2009, depending on the shape of the input data and the number of components to extract. [24]

3) Classification functions:

- `flatten_and_normalize(data)`: Given an array of data, this function reshape the data (flattening it), and uses Scikit-learn's *MinMaxScaler* class to normalize it (between 0 and 1).
- `split_data(data, train_size, test_size)`: Given an array of data, this function uses Scikit-learn's `'train_test_split'` function to split the audio spectral features data randomly into a training set and a validation set.
- `svm_classification(gamma_arg, kernel_arg, boosting)`: This function uses the spectral features, that were split into a training set and a validation set, to train an SVM classification model and to predict results this model. This function uses Scikit-learn's *SVM* class.
- `svm_classification(gamma_arg, kernel_arg, boosting)`: This function is similar to `'svm_classification(gamma_arg, kernel_arg, boosting)''`, but uses the training and validation sets post-PCA transformation.
- `random_forest_classification(depth, number_of_trees, bootstrap)`: This function uses the spectral features, that were split into a training set and a validation set, to train a *Random Forest* classifier model, and to predict results for this model. This function uses scikit-learn's *RandomForestClassifier* class.
- `random_forest_classification_pca(depth, number_of_trees, bootstrap)`: This function is similar to textttrandom_forest_classification(depth, number_of_trees, bootstrap), but uses the training and validation sets post-PCA transformation.
- `adaboost_classification_pca(n_estimators, learning_rate, algorithm)`: This function uses the post-PCA training set and the validation set, to train an AdaBoost classifier model(with Decision Trees as the base estimator), and to predict results this model. This function uses scikit-learn's *AdaBoostClassifier* class.

IV. RESULTS

A total of three classes were selected for our preliminary experiment. The number of songs represented by these classes

was 237. The artists selected had substantially different musical styles but all fell inside of the rock genre. An permutation approach was followed to test every possible combination of parameters in order to attempt to find the best ones. In SVM both boosting and not boosting were implemented. Gamma values ranging from 50 to 1000 were tried. Four different kernel types (rbf, poly, linear and sigmoid), provided by the library, were also implemented. In random forrest, depth and number of trees were also changed algorithmically providing a large number of trials. Both bootstrapping and no boostrapping were tried for all different values of depth and *ntrees*. A similar iterative process was explored with the AdaBoost classifier. While the results of our priliminary analysis were positive, with Spectral Contrast and random forest providing an accuracy of 81%, we had difficulties scaling the project to work with the entire set. While we were able to run our system using all 20 classes, we are not convinved that the results we got were correct. Before reporting these results, we expect to be able to retrain our model and test the validity of these results as they seem to be at odds with what is possible using these algorithms. It is possible that in our attempt to maximize performance we might have created models that overfit or underfit our data. That being said, it seems unlikely that this would be the case for all possible parameters selected. More testing must be conducted in order to understand why our model failed using the larger set.

	MFCCs	MFCCs + PCA	Spectral Contrast	Spectral Contrast + PCA	MFCCs + Spectral Contrast	MFCCs + Spectral Contrast + PCA
SVM	45.833%	47.916%	79.166%	64.583%	62.5%	58.333%
AdaBoost	-	58.333%	-	72.916%	-	54.166%
Random Forest	54.166%	56.25%	81.25%	70.833%	60.416%	54.166%

Fig. 3. Results for classifying 3 classes

V. Future Work

Some previous works performed with this dataset called for a 6-fold cross validation cut model which we did not implement. Our system simply took random samples from all the data and trained the model using that. It would be interesting to see just how much splitting the data the way the creators of the database suggest can affect the results. Secondly, we need to evaluate the system numerous times. Unfortunately the results we report were not cross-validated and are the result of a single trial. That being said using our approach we now have a good idea of what values to use for all our parameters and perhaps which ones might be redundant, allowing us to concentrate of testing using those parameter values. We are also curious about how exactly vocal segmentation can be accomplished and are interested in using those techniques in our future research. Finally, and this is will be our very first step, we have to run our model again with the entire set, as is, and see if the results we found are outliers or real values. If the same results are found we will have to think about training a model for each class or seeing what the results are when running 5, 10 and 15 classes.

VI. Discussion

Artist recognition is not a simple task. In contrast to popular MIR algorithms that can only match songs which our found in the database, artist recognition algorithms attempt to predict artists by using prior knowledge. The relevance of these algorithms can be extrapolated from their use in copyright law, musicology and database rehabilitation. Much research has been done in the field and research has shown that good matching can be accomplished. That being said, when compared to human ability for artist recognition, these system's still have ways to go. It is only by continuing to learn and understand how these systems work that we will be able to obtain the accuracy that we seek. We hope that we can be part of the innovative people in the field that push the research forward and find more ways in which these techniques can be used in positive ways.

References

[1] Adam L Berenzweig, Daniel PW Ellis, and Steve Lawrence. Using voice segments to improve artist classification of music. In *Audio Engineering Society Conference: 22nd International Conference: Virtual, Synthetic, and Entertainment Audio*. Audio Engineering Society, 2002.

[2] James Bergstra, Norman Casagrande, Dumitru Erhan, Douglas Eck, and Balázs Kégl. Aggregate features and adaboost for music classification. *Machine learning*, 65(2-3):473–484, 2006.

[3] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[4] Judith C Brown. Computer identification of musical instruments using pattern recognition with cepstral coefficients as features. *The Journal of the Acoustical Society of America*, 105(3):1933–1941, 1999.

[5] Ivan Bruha and A Famili. Postprocessing in machine learning and data mining. *ACM SIGKDD Explorations Newsletter*, 2(2):110–114, 2000.

[6] Chih-Chung Chang and Chih-Jen Lin. Libsvm: a library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, 2(3):27, 2011.

[7] David Cope. Computer modeling of musical intelligence in emi. *Computer Music Journal*, 16(2):69–83, 1992.

[8] R Dony et al. Karhunen-loeve transform. *The transform and data compression handbook*, 1:1–34, 2001.

[9] Shlomo Dubnov, Gerard Assayag, Olivier Lartillot, and Gill Bejerano. Using machine-learning methods for musical style modeling. *Computer*, 36(10):73–80, 2003.

[10] Bradley Efron and Robert J Tibshirani. *An introduction to the bootstrap*. CRC press, 1994.

[11] Daniel PW Ellis. Classifying music audio with timbral and chroma features. In *ISMIR*, volume 7, pages 339–340, 2007.

[12] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001.

[13] Trevor Hastie, Saharon Rosset, Ji Zhu, and Hui Zou. Multi-class adaboost. *Statistics and its Interface*, 2(3):349–360, 2009.

[14] Dan-Ning Jiang, Lie Lu, Hong-Jiang Zhang, Jian-Hua Tao, and Lian-Hong Cai. Music type classification by spectral contrast feature. In *Multimedia and Expo, 2002. ICME'02. Proceedings. 2002 IEEE International Conference on*, volume 1, pages 113–116. IEEE, 2002.

[15] Xin Jin and Rongfang Bie. Random forest and pca for self-organizing maps based automatic music genre discrimination. In *Conference on Data Mining— DMIN*, volume 6, page 415, 2006.

[16] Tryphon Lambrou, Panos Kudumakis, R Speller, M Sandler, and A Linney. Classification of audio signals using statistical features on time and wavelet transform domains. In *Acoustics, Speech and Signal Processing, 1998. Proceedings of the 1998 IEEE International Conference on*, volume 6, pages 3621–3624. IEEE, 1998.

[17] Beth Logan et al. Mel frequency cepstral coefficients for music modeling. In *ISMIR*, 2000.

[18] Michael I Mandel and Dan Ellis. Song-level features and support vector machines for music classification. In *ISMIR*, volume 2005, pages 594–599, 2005.

[19] Alan V Oppenheim and Ronald W Schafer. From frequency to quefrency: A history of the cepstrum. *IEEE signal processing Magazine*, 21(5):95–106, 2004.

[20] Douglas A Reynolds. Experimental evaluation of features for robust speaker identification. *IEEE Transactions on Speech and Audio Processing*, 2(4):639–643, 1994.

[21] Joan Serra, Emilia Gómez, and Perfecto Herrera. Audio cover song identification and similarity: Background, approaches, evaluation, and beyond. *Advances in Music Information Retrieval*, 274:307–332, 2010.

[22] Stanley Smith Stevens, John Volkmann, and Edwin B Newman. A scale for the measurement of the psychological magnitude pitch. *The Journal of the Acoustical Society of America*, 8(3):185–190, 1937.

[23] Johan AK Suykens and Joos Vandewalle. Least squares support vector machine classifiers. *Neural processing letters*, 9(3):293–300, 1999.

[24] Michael E Tipping and Christopher M Bishop. Mixtures of probabilistic principal component analyzers. *Neural computation*, 11(2):443–482, 1999.

[25] Michael E Tipping and Christopher M Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 61(3):611–622, 1999.

[26] Hal Varian. Bootstrap tutorial. *Mathematica Journal*, 9(4):768–775, 2005.