Learning Instrument Identification

Lewis Guignard and Greg Kehoe
Stanford University
lewisg@stanford.edu gpkehoe@stanford.edu

Abstract

In this project, we utilize machine learning techniques to construct a classifier for automatic instrument recognition given a mono-track audio recording. We focus on the classification of eight instruments commonly used in rock music bands. By examining the spectral content of each instrument, we propose a set of features that can be used to accurately classify musical instruments and we present results from both supervised and unsupervised learning algorithms applied to our generated data set.

1. Introduction

The primary goal of this project is to classify single-instrument recordings for eight instruments: Acoustic Guitar, Electric Guitar, Electric Bass Guitar, Tenor Drum, Bass Drum, Snare Drum, Cymbals, and Hi-Hat. The motivation for instrument identification is to provide insights and results that we hope to use in our continued research of multiple-source separation from a single mono-track recording.

The input to our model consists of individual audio recordings for each instrument in MP3 format. We perform feature extraction from the discrete fourier transform (DFT) of the normalized signal and label each example with a unique integer corresponding to the instruments' class. We then use this data for analysis and train multiple SVMs to make predictions on the test set.

2. Related Work

Recent work on instrument classification for music information retrieval (MIR) focuses on designing feature sets to accurately identify instruments on a monophonic or polyphonic audio recording. Popular approaches include manual feature set construction and evaluation using classic machine learning algorithms [1], [2], the use of neural networks to learn the feature set [3], and combinations of manual design techniques with neural networks for learning [4]. In each of these studies a common goal is to create or learn a set of features that will improve classifier accuracy. Additionally, a small set of features are typically derived from the DFT such that computational complexity is reduced.

This project contributes a novel feature set design that allows us to obtain 93.1% classification accuracy using an SVM. Similar to other work, we manually designed the feature set from the DFT. However, we do not use features extracted from the time-domain signal.

3. Dataset

Our data set consists of 1,455 samples, obtained from both online and live recordings. The table below shows the distribution of samples. We originaly sought to use samples from entirely online sources for rapid data collection. However, individual samples for guitar proved difficult to find. In order to generate samples for each guitar, we used an AudioBox USB and Audacity to record individual notes along the fretboard of each guitar and generate MP3 files with a duration of approximately 3-4 seconds per note. Samples for the percussion instruments were extracted from YouTube videos using [5].

Instrument	n Samples	% of Data
Acoustic Guitar	146	10.0
Electric Guitar	362	24.9
Electric Bass Guitar	270	18.6
Tenor Drum	153	10.5
Bass Drum	125	8.6
Snare Drum	156	10.7
Cymbals	127	8.7
Hihat	116	8.0

Given that our Feature Selection depends

on the spectral content of the training examples, we were concerned about how dissimilarites in quality between WAV audio, easily produced by our live recordings, and MP3 audio as output by [5] would bias the data set. A brief analysis of the DFT for the same recording encoded as both 16-bit WAV at 44.1 KHz and MP3 at 32 kbps revealed that no significant information loss occurred that incumbured our feature extraction. Therefore, we decided to use MP3 as the common input format.

4. Features

Features for each sample are obtained by using the DFT and identifying the 10 most predominant frequencies with the greatest percentage contribution to the total power of the signal. Given that our samples are not uniform in length, we first normalize each sample to have unit energy. This also removes any effects due to variance in volume levels among the samples. After normalizing, we find the 10 frequencies with greatest amplitude and integrate over a log range centered at each frequency to compute the power contribution. Using a log range accounts for a larger octave band [6] for notes at higher frequencies and models our assumption that the guitars are tuned using twelve-tone equal temperament [7]. Thus, for each of the 10 selected frequencies, we extract a 3-tuple of frequency, power, and amplitude such that the nth tuple represents the frequency component with the nth largest power contribution. An example of feature extraction is shown in Figure 1 for the same note played on the bass guitar and acoustic guitar. We observe that each sample gives a unique signature of each instrument in the frequency domain.

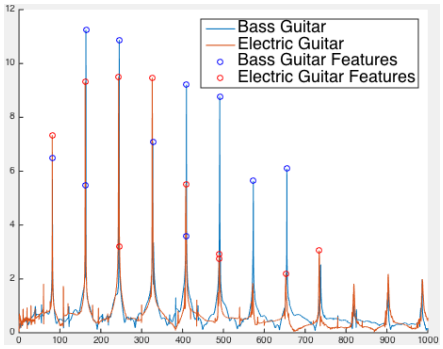

Figure 1: *Feature Selection, Bass and Electric Guitar*

Alternatively, we considered using the entire FFT as the feature vector for each sample. However, in order to avoid high variance, we chose to select a small number of discrete features relative to the size of our training set. This stemmed primarily from the difficulty of acquiring a large number of training examples. Furthermore, by selecting a small set of features shown to be most relevant by our preliminary analysis, we were able to obtain reduced computational complexity and achieve run times of under a minute.

5. Methods

Several models were applied to the dataset, first in an exploratory manner (K-Means, Principal Component Analysis (PCA)), and then for instrument classification, a Support Vector Machine implementing a variety of kernels.

5.1. K-Means

K-Means lends itself as a quick model to attempt on the data, and as its a relatively simple model to implement, gave us good intuition as to how separable the classes are, at least in a linear regime.

In the K-Means classification algorithm, K centroids are assigned randomly in the space of the data, centroids being the Euclidean centers of mass for each cluster. In our case, K = 8 for all the instruments, but we also clustered on lower values of K to look at subsets of instruments. Once the centroids are randomly assigned, the following algorithm is iterated until centroid locations stop changing:

1. Assign each example to its closest centroid
2. Update centroids to be the center of mass for the examples assigned to it.

After the centroid locations stop updating, the algorithm has completed (converged), and K classes have been assigned to each datapoint. This unsupervised learning procedure can then be compared to our instrument classes to get an idea of how well each instrument is separated in the feature space chosen.

5.2. Principal Component Analysis

PCA is a method of dimensionality reduction that can be used for visual inspection of the data, and can also be used to map large dimensional data to smaller dimensions (feature sizes), while capturing as much of the variance (information) of the data as possible. For our purposes, PCA was used to visually interpret any structure in the data, and again gain an intuition of both how separable each class is, and what models might perform better than others in separating them.

PCA uses a Singular Value Decomposition to find the principal eigenvectors of the data. One can then visualize the data in this new space, using a subset of the first n principal components, ordered by amount of variance of the data explained. 'n' is usually picked as 2 or 3 for visualization purposes. Finding the direction of the first principal component is equivalent to finding u such that $||u||^2 = 1$ and:

$$\frac{1}{m}\sum_{i=1}^{m}(x^{(i)T}u)^2$$

is maximized, where m is the number of training examples, and $x^{(i)}$ is a specific training example. The second principal component can then be found by finding the direction v that maximizes the same equation, but is orthogonal to u, and so on for higher order principal components. In this way, one can be sure to describe the maximum amount of variance of the data in the fewest directions.

The mean and variance of each feature are usually normalized (to 0 and 1, respectively) when performing PCA, so as not to give disproportionate importance to specific features.

5.3. Support Vector Machine

Support Vector Machines (SVMs) are one of the most popular 'out of the box' machine learning algorithms for classification, for their ease of use and wide success. The SVM works by finding an optimal hyperplane separating the data into two classes, that gives the largest 'geometric' margin, i.e. has all data maximally distant from the hyperplane. When data is not linearly separable, one can include Lagrange Multipliers that introduce error terms that allow some examples to be inside the geometric margin, and even possibly on the wrong side of the hyperplane. The maximization problem is:

$$max_\alpha \sum_{i=1}^{m}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{m} y^{(i)}y^{(j)}\alpha_i\alpha_j \left\langle x^{(i)}, x^{(j)} \right\rangle \quad (1)$$

$$s.t. 0 \leq \alpha_i \leq C, i = 1, ..., m \quad (2)$$

$$\sum_{i=1}^{m}\alpha_i y^{(i)} = 0 \quad (3)$$

These α_i and inner products can then be used to solve for the hyperplane:

$$\omega^T x + b = \sum_{i=1}^{m}\alpha_i y^{(i)} \left\langle x^{(i)}, x \right\rangle + b$$

With this theoretical motivation, one can use the Sequential Minimal Optimization algorithm to solve for the hyperplane of a given dataset.

One of the great benefits of SVM is the inner product of datapoints in the optimization. This can be replaced by a Kernel, which can map the features into a higher dimensional space, including mixing terms, etc.

When multiple classes are present, SVM can still be used. There are two ways to tackle the problem, namely one versus one or one versus all. In one versus one, there are $K(K-1)/2$ models trained, one for each pair of classes (where K is the number of classes). The final classification of a test point is given to the class for which it gets the most votes from the above models.

5.4. Regularization and Model Selection

For regularization, we used 10-fold cross validation (CV) to compare different models used. 10-fold cross validation trains on 90 % of the data and tests on the remaining 10 %, and repeats for all 10 ways to divide the dataset in this way. The test error from each hold-out set is then averaged to find the estimated test error of the model overall. In this way, one can still train on the whole dataset given, and achieve a pseudo test error.

In choosing which features to use in the model, we implemented both forward search and backward search and compared the results. In forward search, one trains a model on each feature alone and chooses the model with the lowest CV error. Then, one iteratively checks which of the remaining unused features will lower the CV error when included in the model. The algorithm finishes when adding any unused feature will only increase the CV error. In backwards search, one starts with a model using all features, then iteratively chooses to remove one feature that lowers the CV error the most. The algorithm terminates when removing any more single features will only increase the CV error.

6. Results / Discussion

6.1. K-Means

In implementing K-Means on all eight instruments, we found no distinguishable mapping from the classes generated by the clustering to the classes of instruments in our dataset. We assume this is because, at least in a linear space, there are large overlaps of features from each class. We then asked ourselves how well K-Means would work on instruments whose fourier transforms seemed far apart. We measured success of the algorithm by ratio of number of correctly classed datapoints to total number of datapoints.

In comparing the data of Electric Guitar and Cymbals, K-Means with 2 clusters achieved 93.2 % accuracy, and when comparing Electric Guitar to Tenor Drum, 58.4% accuracy was obtained. With no good clustering on all eight instruments, or on separating the Electric, Bass, and Acoustic Guitar, we moved to other algorithms.

6.2. Principal Component Analysis

The goal of running PCA on the dataset was exploratory in nature; to look for any structure in the dataset and gain insight into the separability of the classes of instruments. We found when looking at all instruments in the first three components, a tetrahedral shape, where the guitars were spread along one face, and cymbals along another. All data converges to one point in this space, and three boundaries of the tetrahedron are sharp, in that data does not move beyond them. We would be interested in finding other instruments or sounds / effects that might cross these boundaries.

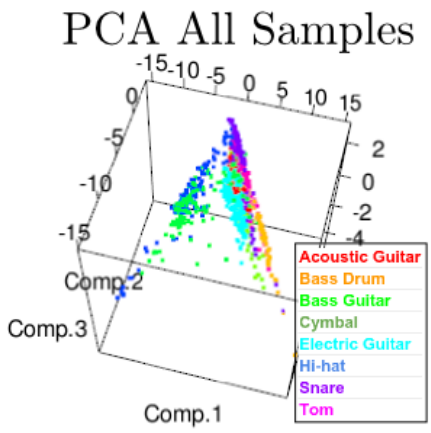

Figure 2: *PCA of all data*

Figure 2 shows the complete dataset in the first three principal components, explaining 66% of the variance of the data, with proportion of variance explained for each component being:

PC #	1	2	3
% Var expn'd	38.1	21.4	6.8

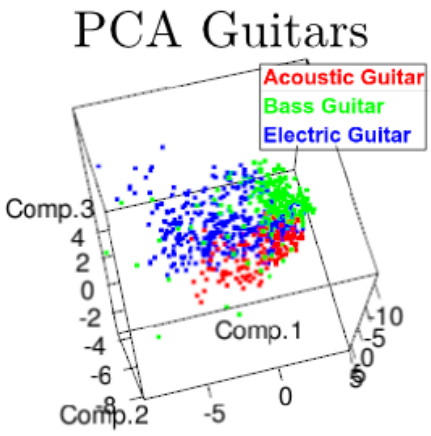

Figure 3: *Initial Feature Selection*

Figure 3 illuminates the three guitars alone in the first three principal components. The instruments are not linearly separable in this

subspace, but their respective centroids are distinct. We find the mediocre results of K-Means surprising, considering how distinct the data is here, although we note that regularization or model selection was not performed on the feature set for K-Means. The three principal components explain 53.9% of the variance, with the proportion in each component being:

PC #	1	2	3
% Var expn'd	26.3	18.4	9.2

6.3. Support Vector Machine

For the classification algorithm, we chose an SVM algorithm to attempt to classify all 8 instruments, and to classify the subset of Electric, Acoustic and Bass Guitars. Models of Linear, Gaussian, and Polynomial kernels were trained on a random 80 % subset of the data, and tested on the remaining 20% for a range of applicable parameters. We found the polynomial kernel to have the best performance of the three kernels, varying cost (C), Gamma (γ), and degree (d) per the below kernel (for cost, see equation (2)).

$$K(x^{(i)}, x^{(j)}) = (\gamma x^{(i)T} x^{(j)})^d \tag{4}$$

We attempted a model for every combination of the following values of parameters:

$$C \in \{.1, 1, 10, 100\}$$
$$\gamma \in \{.5, 1, 2\}$$
$$d \in \{2, 3, 4\}$$

The model with lowest test error when classifying all instruments used: $C = .1, \gamma = 1, d = 2$, and when classifying the three guitar instruments, used: $C = .1, \gamma = .5, d = 3$.

Using these optimal parameters, we ran 10-fold cross validation with each model on its respective dataset, then applied both forward and backward selection to attempt a better error rate using a subset of the features.

	All instruments	Guitars
CV Error	17.8%	13.1%
Forward	17.3%	8.9%
Backward	15.4%	6.9%

Using the backwards subset of the features (29 of the 30 features), looking only at guitars, we train on 80% of the data and test on the remaining 20% to produce the following confusion matrix.

	Truth			
Predict		Acou	Bass	Elec
	Acou	23	1	0
	Bass	0	43	2
	Elec	1	3	76

7. Conclusions / Future Work

When separating musical instruments, choosing an indicative subset of features to predict on is extremely important. Using frequency, amplitude and power of the top ten peaks of the spectrum of the samples proved a good subset of features, as shown in PCA visually and SVM analytically. K-Means did not perform well, and we find this intriguing considering how well SVM performed. We assume this is because of the freedom that the SVM method has in mapping to higher order spaces, and including mixing terms.

Given more time and resources, we would move onto source separation, possibly augmenting an Independent Component Analysis method, or exploring neural networks.

References

[1] Y Takahashi and K Kondo. Comparison of two classification methods for musical instrument identification. In *Consumer Electronics (GCCE), 2014 IEEE 3rd Global Conference on*, pages 67–68. IEEE, 2014.

[2] Elzbieta Kubera, Alicja A Wieczorkowska, and Magdalena Skrzypiec. Influence of feature sets on precision, recall, and accuracy of identification of musical instruments in audio recordings. In *ISMIS*, pages 204–213. Springer, 2014.

[3] Peter Li, Jiyuan Qian, and Tian Wang. Automatic instrument recognition in polyphonic music using convolutional neural networks. *arXiv preprint arXiv:1511.05520*, 2015.

[4] DG Bhalke, CB Rama Rao, and DS Bormane. Automatic musical instrument classification using fractional fourier transform based-mfcc features and counter propagation neural network. *Journal of Intelligent Information Systems*, pages 1–22, 2015.

[5] Youtube to mp3 generator. `www.youtube-mp3.org`.

[6] Octave band. `https://en.wikipedia.org/wiki/Octave_band`.

[7] Equal temperament. `https://en.wikipedia.org/wiki/Equal_temperament`.