Campus Location Recognition using Audio Signals

James Sun,Reid Westwood
SUNetID:jsun2015,rwestwoo
Email: jsun2015@stanford.edu, rwestwoo@stanford.edu

I. Introduction

Recognizing one's location by sound is a coarse skill that many people seem to develop out of routine. We may be able to recognize a favorite café by the genre of music playing and the baristas' voices. We may be able to recognize the inside of our car by the noises coming out of the engine and chassis. We might come to associate the sounds coming through our rooms' windows with home. However, are these sounds by themselves truly sufficient to identify the locations that we frequent? This project attempts to answer that question by developing a Machine Learning system that recognizes geographical location purely based on audio signal inputs. To emulate a typical Stanford student, the system is trained on sounds at locations along a path that a student might take as he or she goes about a typical school day. In the process of developing this system, we investigated audio features in both the spectral and time domain as well as multiple supervised learning algorithms.

II. Related Work

A previous CS229 course project identified landmarks based on visual features [1]. [2] gives a classifier that can distinguish between multiple types of audio such as speech and nature. [3] investigates the use of audio features to perform robotic scene recognition. [4] integrated Mel-frequency cepstral coefficients (MFCCs) with Matching Pursuit (MP) signal representation coefficients to recognize environmental sound. [5] uses Support Vector Machines (SVMs) with audio features to classify different types of audio.

III. Scope

As stated in Section I, we have limited the number of areas that the system will recognize. Furthermore, we have limited the geographical resolution of labels to named locations encompassing areas such as Rains Graduate Housing. Both of these limitations are in line with how a typical person may use audio cues to identify his or her location. As such, these geographical restrictions in scope are unlikely to be relaxed.

We have also initially limited our scope temporally to data gathered on weekends in the Spring Academic Quarter. Initial results are promising, and we plan to gather data during the weekdays as well.

IV. System Design

A. Hardware and Software

The system hardware consists of an Android phone and a PC. The Android phone runs the Android 6.0 Operating system and uses the `HI-Q MP3 REC (FREE)` application to record audio. The PC uses Python with the following open-source libraries:

- Scipy
- Numpy
- statsmodels
- scikits.talkbox
- sklearn

The system also makes use of a few custom libraries developed specifically for this project.

B. Signal Flow

The following details the flow of a signal when making a prediction

1) Audio signal is recorded by the Android phone
2) Android phone encodes the signal as a Wav file
3) The Wav file enters the Python pipeline as a `Sample` instance
4) A trained `Classifier` instance receives the `Sample`
   a) The `Sample` is broken down a number of subsamples based on a predetermined audio length for each subsample
   b) A prediction is made on each subsample
   c) The most frequent subsample prediction is output as the overall prediction.

A graphical illustration of this is shown in Figure 1:

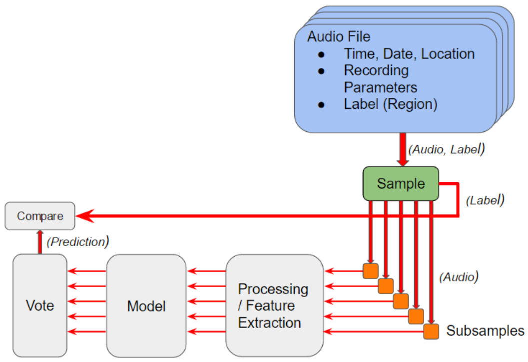


Fig. 1: System Block Diagram

C. Locations

The system is trained to recognize the following 7 locations:

0. Arrillaga Gym
1. Bytes Café
2. Circle of Death
   Intersection of Escondido and Lasuen
3. Huang Lawn
4. The Oval
5. Rains Graduate Housing
6. Tressider Memorial Union

These locations represent the route a typical graduate engineering student living at Rains might take on a typical day. Locations 0,1, and 6 are indoors whereas Locations 2,3,4, and 5 are outdoors.

V. Data Collection

A. Audio Format

Data is collected using the `HI-Q MP3 REC (FREE)` application as noted in Section IV-A. This application is freely available on the Google Play Store. Monophonic Audio is recorded without preprocessing and postprocessing at a sample rate of 44.1 kHz.

B. Data Collection

Initial training data was over a period of 2 days. Data was gathered at each location an equal number of times. Each data collection event followed the following procedure:

1) Configure `HI-Q MP3 REC (FREE)` to record audio as in V-A.
2) Hold the Android recording device away from body with no obstructions of the microphone
3) Stand in a single location throughout the recording
4) Record for 1 minute
5) Throw away recording if person recording interferes with the environment in some way (talks to a bystander, causes a bicycle crash, heckles passerby, etc...)
6) Split recording into 10-second-long samples

In total, we gathered 251 recordings of 1 minute in length, for a total of 1506 data samples of 10 seconds in length.

VI. Audio Features

We investigated the use of the following features:

- Mean Amplitude in Time Domain
- Variance of Amplitude in Time Domain
- Fourier Transform (40 bins)
- Autocorrelation (ACF) (40 bins)
- SPED (60 bins)
- 13 Mel-frequency cepstral coefficients (MFCCs)

We observed best performance using MFCC and SPED features for a total of 73 features. These 2 feature types are described in the subsequent subsections.

A. MFCC

MFCCs are commonly used to characterize structured audio such as speech and music in the frequency domain, often as an alternative to the Fourier Transform [3], [4], [5], [6]. Calculating the MFCCs proceeds in the following manner [7]:

1) Divide the signal into short windows in the time domain
2) For each windowed signal:
   a) Take the Fast Fourier Transform (FFT)
   b) Map powers of the FFT onto the Mel scale (which emphasizes lower frequencies)
   c) Take the logarithm of the resultant mapping
   d) Take the discrete cosine transform (DCT) of the log mapping at a certain number of frequencies
   e) Output a subset of the resulting DCT amplitudes as the MFCCs

We used 23.2 ms windows and kept the first 13 MFCCs as is standard [4]. This creates multiple sets of MFCCs per signal (one per window). To summarize all of these coefficients, we take the mean over all windows of a signal.

B. SPED

SPED, Subband Peak Energy Detection, is a method of finding consistent sources of energy (in frequency) over time. First, a spectrogram is generated using time-windowed FFTs on the time-domain signal. The result is the energy of the signal as a function of both time and frequency. SPED then finds the peaks across frequency as defined by some window size.

A local maximum is marked '1', and all other elements are zero. Finally, this matrix is summed across time to give a rough histogram of local maxima as a function of frequency. Finally, because of the fine resolution of the FFT in frequency, we use bin the results according to a log scale.

The idea behind this method is to find low-SNR energy sources that produce a coherent signal. For example, a motor or fan may produce a quiet but consistent sum of tones. In an FFT, this may or may not be visible. However, it will likely result in local maxima over time. Since all maxima are weighted equally, SPED seeks to expose all consistent frequencies regardless of their power. Below, we show a SPED output for Bytes Café and Arrillaga Gym across different days and different areas.

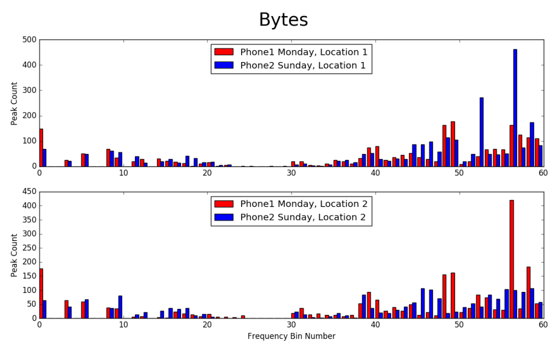


Fig. 2: Sample SPED at Bytes

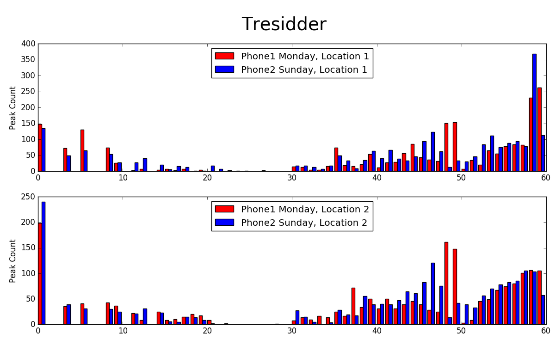


Fig. 3: Sample SPED at Tresidder

C. *Principal Component Analysis*

We investigated the appropriateness of our features by doing a Principal Component Analysis (PCA) on our data set using the above features. Figure 4 plots the fraction of variance explained vs the number of principal components used. We saw that the curve is not steep, and most likely over 50 of our 73 features do in fact encode significant information.

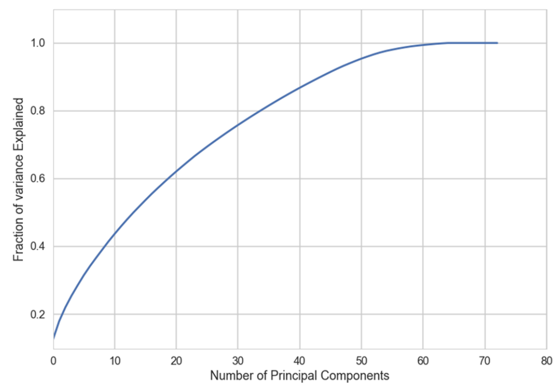


Fig. 4: Variance Explained Vs # of Principal Components

We also projected our samples onto the basis defined by the first 3 principal components for visualization. Certain regions were clearly separable using just these 3 components, such as in Figure 5. Other regions were not quite so obviously separable, as shown in Figure 6

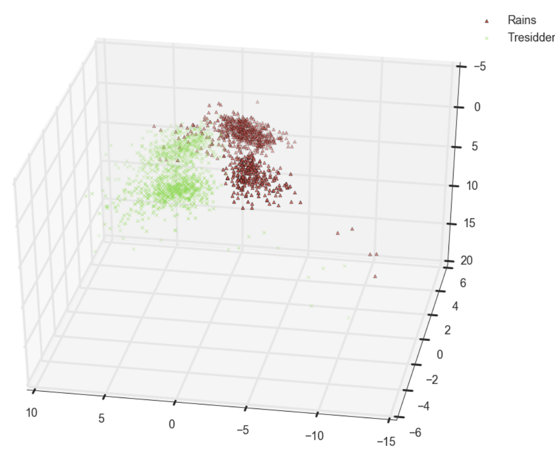


Fig. 5: Rains vs Tressider using the first 3 PCs

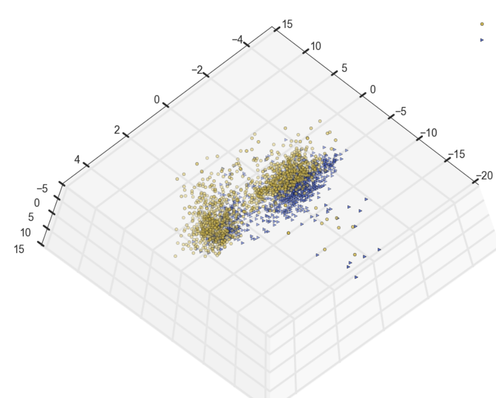


Fig. 6: Oval vs Circle using the first 3 PCs

VII. Methods and Results

Using the MFCC and SPD features, we investigated the following classifiers:

- SVM using Gaussian and Linear Kernels
- Logistic Regression
- Random Forest
- Gaussian Kernel SVM with Logistic Tiebreaker

using Logistic Regression, SVM with a linear kernel, and SVM with a radial basis kernel function, experimenting with different subsets of features with each algorithm. In reference to the system flow in Section IV-B, we choose 10-second samples and 2-second subsamples. So, a sample will be input to the system, 5 predictions (1 for each subsample) will be made, and finally the most frequent prediction will be output. For training, we randomly shuffle the samples and use $2/3$ as training and $1/3$ as testing. To make valid comparisons across different training instances, we reset to a given random number generator seed whenever shuffling the samples. The training flow diagram is similar to the prediction flow diagram:

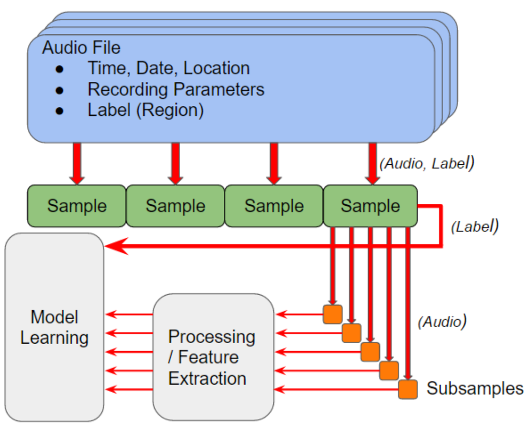
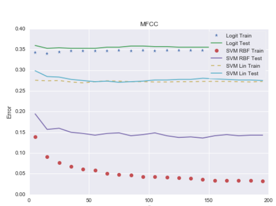
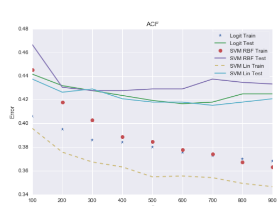
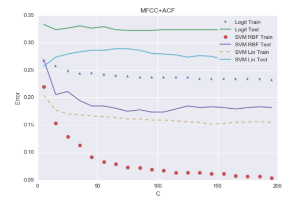


Fig. 7: Training System Block Diagram

We train each classifier on subsamples. So, for each training sample, we generate 5 training subsamples. However, we calculate training and testing error based on entire samples, using the aforementioned procedure. Some of the results are below with different error penalty terms C:







From these results, we conclude that the most promising simple classification algorithm is using MFCC features to train an RBF-kernel SVM with a penalty parameter $C = 15$, despite the considerable gap between training and testing errors. Surprisingly, adding features in conjunction with MFCC features does not seem to improve performance for any of the simple classifiers we tried.

Note we also tested other features as well that due to poor performance have not been plotted here. We saw that the FFT features did not improve upon performance when combined with the ACF features. In retrospect, this somewhat makes sense because the ACF can be calculated directly from the FFT. Somewhat surprisingly, we saw extremely bad performance when using just FFT features. For some reason, all learning algorithms would classify all inputs as Bytes Café. This may merit further investigation later.

Also, we saw that the SPED features we calculated worked reasonably well when used in conjunction with a Linear SVM, obtaining about 20% test error. However, it performed quite poorly using the RBF-kernel SVM that worked well with the MFCC features. In order to incorporate both, we will explore the possibility of 'Mixture of Experts' to intelligently combine the two. Furthermore, calculation of the SPED features is still rather inefficient. By exercising feature analysis, we may be able to determine certain frequency ranges that are most useful in order to reduce computations and well as feature size.

VIII. Future Work

We plan on gathering more data points at different days of the week in order to evaluate better the generalization of our system. We also plan to investigate ensemble methods in order to combine features with MFCC in a way that improves performance rather than degrades it. If we cannot force our current feature selection to work well together, we may evaluate some of the other features that have been mentioned in the Literature, including time domain features.

References

[1] A. Crudge, W. Thomas, and t. . Kaiyuan Zhu.

[2] L. Chen, S. Gunduz, and M. T. Ozsu, "Mixed type audio classification with support vector machine," in *2006 IEEE International Conference on Multimedia and Expo*, July 2006, pp. 781–784.

[3] S. Chu, S. Narayanan, C. c. J. Kuo, and M. J. Mataric, “Where am i? scene recognition for mobile robots using audio features,” in *2006 IEEE International Conference on Multimedia and Expo*, July 2006, pp. 885–888.

[4] S. Chu, S. Narayanan, and C. C. J. Kuo, “Environmental sound recognition with time and frequency audio features,” *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 17, no. 6, pp. 1142–1158, Aug 2009.

[5] G. Guo and S. Z. Li, “Content-based audio classification and retrieval by support vector machines,” *Neural Networks, IEEE Transactions on*, vol. 14, no. 1, pp. 209–215, 2003.

[6] J.-J. Aucouturier, B. Defreville, and F. Pachet, “The bag-of-frames approach to audio pattern recognition: A sufficient model for urban soundscapes but not for polyphonic music,” *The Journal of the Acoustical Society of America*, vol. 122, no. 2, pp. 881–891, 2007.

[7] L. Rabiner and B.-H. Juang, “Fundamentals of speech recognition,” 1993.