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) The audio signal is recorded by the Android phone
2) The 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 into subsamples of 1 second in length
   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:

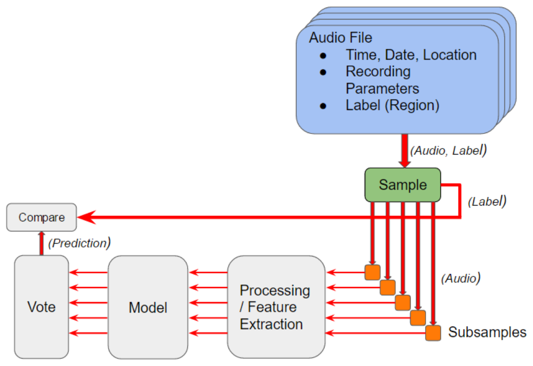


Fig. 1: System Block Diagram

We have designed the system with this subsample structure so that any audio signal with length greater than 1 second can be an input.

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. Even though our system is designed to handle any inputs of length greater than 1 second, we standardized our inputs to be 10 seconds for convenience.

We also attempted to maintain sample balance amongst the 7 locations while also diversifying sample collection temporally. The distribution of samples by location is in Table I. The distribution by day and time is given in Figure 2.

TABLE I: # Samples Gathered at each Location

Rains	Circle	Tressider	Huang	Bytes	Oval	Arrillaga
234	210	211	222	222	192	216

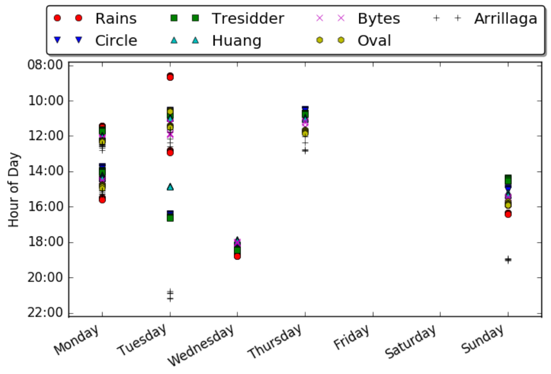


Fig. 2: Sample Distribution by Day

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.

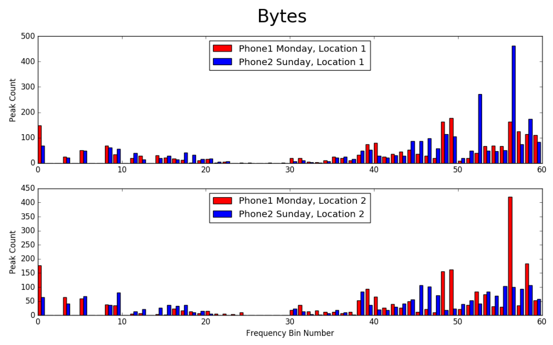


Fig. 3: Sample SPED at Bytes

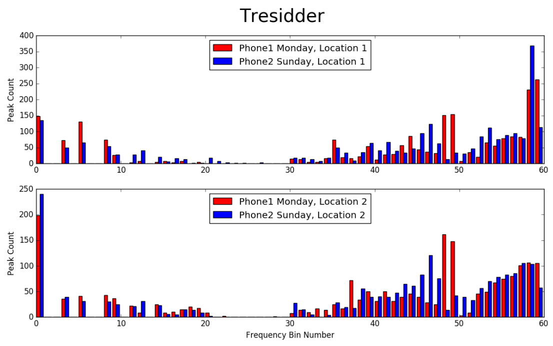


Fig. 4: Sample SPED at Tressider

C. Principal Component Analysis

We investigated the redundancy in our features by doing a Principal Component Analysis (PCA) on our data set using the above features. Figure 5 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.

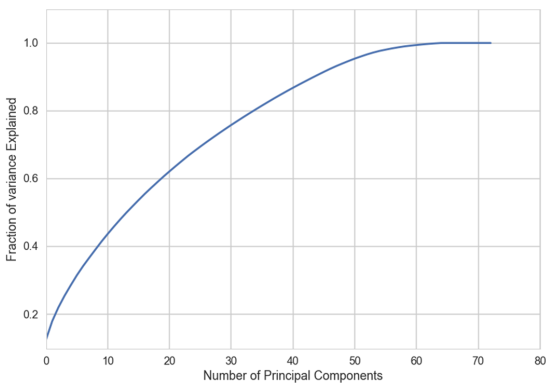


Fig. 5: Variance Explained Vs # of Principal Components

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 6. Other regions were not quite so obviously separable, as shown in Figure 7

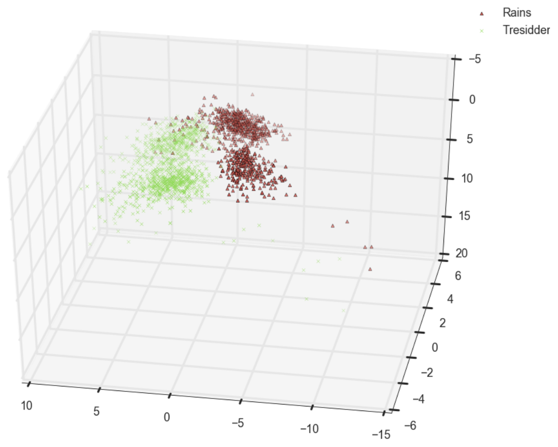


Fig. 6: Rains vs Tressider using the first 3 PCs

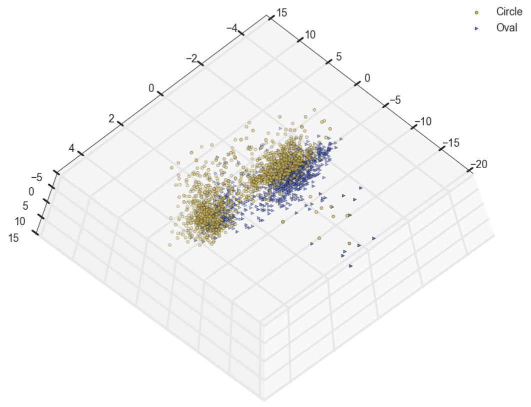


Fig. 7: Oval vs Circle using the first 3 PCs

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

The last classifier uses a Logistic Regression classifier in the case of a tie in the voting process described in Section IV-B. When picking the hyperparameters to use for each classifier, we did a 70%-30% split of our training dataset and then searched over a grid of parameters, evaluating based on accuracy of classification.

A. *Generalization*

We distinguished between 2 types of testing errors:

1) Cross-Validation Error - Error on the testing set when we split the data set completely randomly
2) Generalization Error - Error on the testing set when we split based on random days.

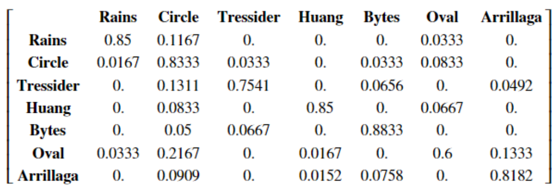
We did this because our data has a significant temporal correlation. We discovered that the typical cross-validation error was too optimistic because audio samples recorded on the same day can be significantly more correlated to each other than to audio recorded on different days. We were able to decrease our Cross-Validation error to around 8% using a Gaussian SVM. However, when we attempt to use this seemingly general classifer on a completely new days' data, we discovered that this classifier was actually very overfitted.

With this aspect in mind, we were able to reduce our Generalization error to a bit less than 20% using a Gaussian SVM with a Logistic Classifier to handle voting ties. To calculate generalization error, we did a form of 5-fold cross-validation. We held out all samples from a single day for testing while using all other days for training, and then we repeat for all 5 days. We finally do a weighted combination to calculate the Generalization Error, weighting based on the number of samples in each held out day. Table II gives a summary of our results.

TABLE II: Classifier Comparison

Classifier	**X-Validation**	**Generalization**
Gaussian Kernel SVM	13.65%	21.72%
Linear Kernel SVM	27.84%	32.74%
Logistic	15.45%	21.22%
Random Forest	14.09%	28.26%
Gaussian SVM + Logistic Tiebreaker	13.89%	19.68%

Using the SVM+Logistic classifier, we generated the following confusion matrix for one of the hold-out trials:



	Rains	**Circle**	**Tressider**	**Huang**	**Bytes**	**Oval**	**Arrillaga**
Rains	0.85	0.1167	0.	0.	0.	0.0333	0.
Circle	0.0167	0.8333	0.0333	0.	0.0333	0.0833	0.
Tressider	0.	0.1311	0.7541	0.	0.0656	0.	0.0492
Huang	0.	0.0833	0.	0.85	0.	0.0667	0.
Bytes	0.	0.05	0.0667	0.	0.8833	0.	0.
Oval	0.0333	0.2167	0.	0.0167	0.	0.6	0.1333
Arrillaga	0.	0.0909	0.	0.0152	0.0758	0.	0.8182

Our classifier did relatively well in terms of accuracy for all but one region: the Oval. However, given that our classifier had high specificity with respect to the Oval label, we posit that this poor accuracy is because our data set had fewer Oval samples in comparison to the other labels (Table I).

B. Classifier Evaluation

As the final step in evaluating our system, we compared the performance of our classifier to people's ability to localize based on audio clips. We created a small game that would present the user with a random 10 second audio sample. The user would then choose from which of the 7 locations the audio was taken; our classifier would do the same. The pool of participants comprised of Stanford CS229 students. The results are shown in Table 8. The sample size was small, with only 41 sample points. However, it seems apparent that even Stanford students, who frequent the chosen locations, are ill-adept at identifying them by sound alone. Of the 41 audio samples, students accurately located only 11 of them for an error rate of 73.2%. This is much higher than our classifier's generalization error of 19.68%. We included our entire data set in the pool of audio samples used by our game, so the recorded performance of our classifier is even better than this generalization error: out of the 41 samples, our classifier correctly classified all but one.

	Rains	Circle	Tressider	Huang	Bytes	Oval	Arrillaga
Rains	1	0	0	2	0	0	0
Circle	1	2	0	2	0	3	0
Tressider	0	0	0	0	0	0	2
Huang	2	3	0	0	0	1	0
Bytes	1	0	2	2	0	1	1
Oval	2	0	0	0	0	4	1
Arrillaga	0	1	0	0	2	1	4

	Rains	Circle	Tressider	Huang	Bytes	Oval	Arrillaga
Rains	0.3333	0.	0.	0.6667	0.	0.	0.
Circle	0.125	0.25	0.	0.25	0.	0.375	0.
Tressider	0.	0.	0.	0.	0.	0.	1.
Huang	0.3333	0.5	0.	0.	0.	0.1667	0.
Bytes	0.1429	0.	0.2857	0.2857	0.	0.1429	0.1429
Oval	0.2857	0.	0.	0.	0.	0.5714	0.1429
Arrillaga	0.	0.125	0.	0.	0.25	0.125	0.5

Fig. 8: Top: Unnormalized Human Confusion Matrix. Bottom: Normalized Human Confusion Matrix

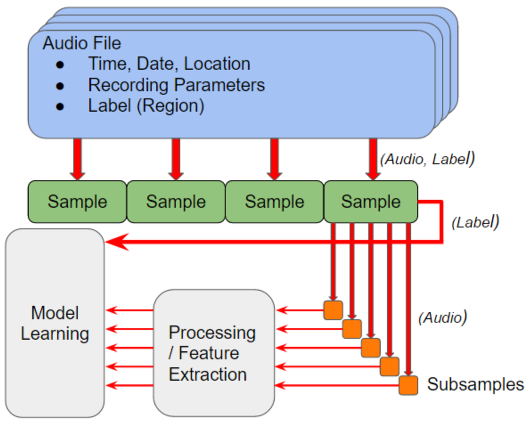


Fig. 9: Training System Block Diagram. MAYBE DELETE THIS ONE?

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.