Mobile Diagnostic Tools for Pulmonary Disease UAP Report

Ryan Lacey

Advisor: Dr. Rich Fletcher
Proposed: March 7, 2014

1 Objective

The purpose of this project is to develop a mobile application for use in clinics with limited resources in developing regions of India. The application will serve as a diagnostic tool for clinicians trained in Ayuverdic medicine to diagnose patients suffering from a pulmonary disease. With an Android mobile device and a low cost digital stethoscope these clinicians will be able to collect patient lung sounds, run diagnostic machine learning algorithms on the lung sound data, and receive classification feedback with the objective of early detection of the onset of pulmonary disease.

2 Motivation

Pulmonary diseases (asthma, pneumonia, lung cancer, tuberculosis, etc.) are an increasing concern in the realm of global health challenges. Chronic obstructive pulmonary disease (COPD) alone is currently the third leading cause of death in the world and second leading cause of death in India. Pneumonia, which also affects the lungs, remains the leading cause of death in Indian children under the age of five. Especially at risk are low income populations due to consequences of living environment, lack of access to health care, and the high cost of screening tools for detection of these diseases.

There is a great need to provide health workers in India a simple tool that can be used to screen for respiratory disease in the primary care setting and identify individuals that require clinical examination and intervention. Since a majority of the clinicians practice Ayuverdic medicine they have little training for diagnosing respiratory problems. As a result many patients go undiagnosed or are misdiagnosed with having a cardiac disease. The disease burden can be significantly reduced if there were better tools that could perform early detection of respiratory disease and enable clinical intervention before the disease became more critical or developed further into chronic disease.

3 Goals

`Phase I`
Utilize machine learning algorithms to classify common disease states.

`Phase II`
Integrate digital monitors into application.

`Phase III`
Develop graphical user interface for Android mobile devices.

Phase I of the project is to develop the tools necessary for patient data to be analyzed. The most fundamental function of the application is for machine learning algorithms to give a binary classification of the data, ie. either the patient has the disease or the patient does not suffer from the disease. Since these classifiers only state "yes or no" instead of "which", each disease will require its own classifier. The most common technique for this type of classification is a simple linear classifier called a support vector machine (SVM). The SVM will train on pre-classified data of Indian patients with the target diseases and aims to minimize potential misclassification of future patients based off of the sample data. Since there are no known machine learning libraries for Android devices a large portion of the work for this project will be in the implementation, training, and testing of these algorithms.

Phase II involves converting audio signals from the digital stethoscope into data for the machine learning algorithms. The stethoscope records data at four locations on the patient's body: the front, side, back, and throat. Lung sound data, as seen in `Figure 1`, will be ran through a pattern recognition algorithm that identifies crackle, wheeze, and plural rub. The output is a two-dimensional matrix incorporating time and sound information from which features can be extracted for the learning algorithm.

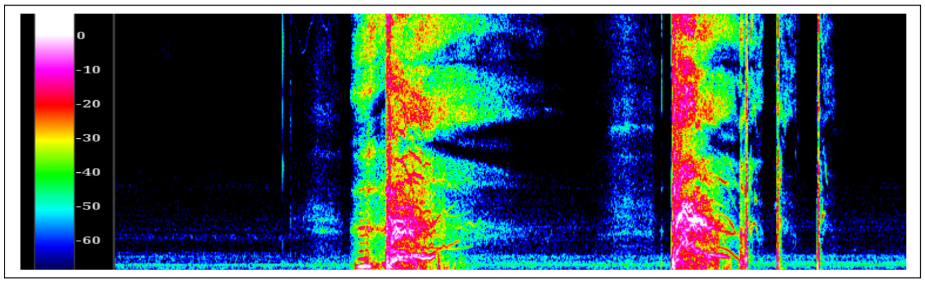

Figure 1: Raw data from lung sounds showing time-frequency plot.

Phase III is the most open ended portion of the project. Development of an intuitive user interface is always an iterative process and is likely to continue as feedback is gathered from the test users and eventually the target clinician group. The first task is to create a skeleton interface that will allow for basic functionality to be realized once the data connection between the monitor and the mobile device is completed. This includes the main view of the application, any menus necessary for operation, and the first steps in graphical displays of patient data. The GUI will be developed in Java using the Android SDK for use on phones and tablets. The ultimate goal is for the interface to be intuitive and language independent so that individuals with limited technological experience will be able to utilize the application.

4 Machine Learning

4.1 Preparation

With the goal of classification in mind, the first task was to determine what tools were to be used on the lung data. I investigated several machine learning libraries available in Java including: Weka, Mallet, and LibSVM. Each of the options, however, had drawbacks that were cause for hesitation. Weka is a comprehensive suite of machine learning algorithms and data mining tools. The software is one of the most popular machine learning packages available and is available into both an executable form to run on pre-formatted data as well as a developers version that can be modified for specific needs. Unfortunately the documentation for the software was rather difficult to traverse and contained a large host of functionalities that both were outside of the scope of the needs of classification and outside of my knowledge in machine learning. Since the timescale for this project was so limited, I wanted to minimize the ramp up time of learning and integrating a complex library, which made this option unattractive. Additionally investigation revealed that potential issues would arise in porting the Weka code to Android, so the long-term use of the software was in question. Mallet was attractive initially because its documentation was more comprehensible and easier to navigate than that Weka provided. However it advertises itself as a package for natural language processing and document classification, which is a large field utilizing techniques not necessarily applicable to the lung sound data. The final consideration, LibSVM, is Java software ported from a popular C package of the same name. Therefore the limited supporting documentation was written in C, which I have no experience with, and had difficulty parsing.

The lack of clear documentation, potential trouble in porting from Java to Android, and concerns over underutilization of the software features at a great cost to system resources called into question the benefit of using a machine learning library. I had implemented classification algorithm's before in a machine learning course I was concurrently enrolled in and pushed forward the idea of doing the algorithms myself for this project. This involved three steps: creating a working classifier, implementing the SVM algorithm, and running classifications on lung data.

My first task was to implement the perceptron algorithm – a simple, but widely used classifier. I had written this before in Python for use in classification of data from Twitter, so conversion to Java was relatively straightforward. The Java implementation was ran on the same Twitter data as a check for correctness, since the performance should have been comparable to the Python code. In regards to the classifier itself the only challenge was performance. The mathematics behind the algorithm is heavily dependent upon multidimensional matrix manipulations. Traditional looping constructs, although viable for correctness, would have yielded a product virtually unusable due to computation time. The Python code addressed this by using the Numpy library. I researched for a comparable linear algebra toolkit in Java and settled on The Apache Commons Mathematics Library. With efficient matricies at hand, I wrote several parsers for formatting the Twitter data and extracting features to classify on. The classifier was tested on a pre-labeled set of test tweets and correctly labeled over 85% of the points. Thus a basic classifier was available for any data set.

The code was structured so that pieces may be swapped out easily. The three main components were the parsers to read in data, algorithms to construct a feature matrix from the data, and the machine learning algorithm to classify the data. The second task was to implement a more powerful classifier – a SVM. The SVM algorithm is similar to perceptron in that they are both linear classifiers. The advantage of the SVM, however, is that it is a maximum margin separator. This means that the classification line will pass through the point that maximizes the distance between the nearest oppositely labeled training data points. If the line passed closer to one of the data points, then the likelihood that the classifier overfits the data increases. Overfitting gives good results on testing data but has the tendency to generalize poorly to test data in the general population.

At this stage SVMs had just been covered in the machine learning course. The techiques for implementing a SVM were very different from the perceptron algorithm and required linear solvers. It became apparent that implementing this myself was not optimal because it would have required yet another library to be brought in to the project and forseeably a great undertaking in working out all of the details of the algorithm. Thus I decided to turn back to incorporating a machine learning library, specifically LibSVM. The modular structure of the existing code made integration of the library's SVM classifier a smooth process with minimal changes. With a SVM in hand the focus of the project switched to classification on the lung data.

(include SVM confidence interval)

4.2 Feature Extraction

Feature extraction is the most open ended and challenging component of the project. Since Android devices have limited processing power due to their mobile processing cores a balance was struck between having enough features to be able to separate lung conditions while keeping computational demand at a minimum. The features should not only correctly classify lung sounds, but also do so with a high degree of confidence.

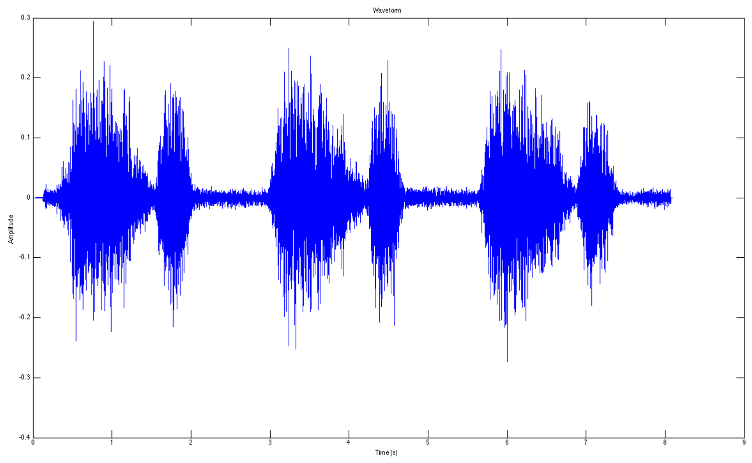

Figure 2: Waveform of vesicular lung sound

Lung data comes in the form of an audio file represented by an of signal amplitude taken at regular intervals determined by the sampling frequency. A graph of amplitude over time represents the waveform of the lung sound, as seen in Figure 2. From the waveform one can see the cycles of inhalation and exhalation, which are the pairs of high amplitude signals separated by low amplitude lulls between breaths.

4.2.1 Wheeze

Lung wheeze is characterized by well-defined pitches over sustained durations. To observe the pitch characteristics the signal is tranformed from the time domain to the frequency domain through a Fast Fourier Transform. Clinicians categorize wheeze into two types: low wheeze and high wheeze. These wheezes, respectively, have large magnitude peaks in low frequency ($<$200 Hz) and higher fequency regions. Figure 3 overlays a vesicular and high pitch wheeze in the frequency domain. The low frequency portions of each signal do not show an appreciable difference in magnitude. The high frequency signal magnitudes, on the other hand, differ greatly. The vesicular signal has a gradual and relatively constant decrease in magnitude moving into the higher frequencies. The wheeze signal instead has a very large magnitude peak that is approximately four times greater than strongest frequency signal of the vesicular data.

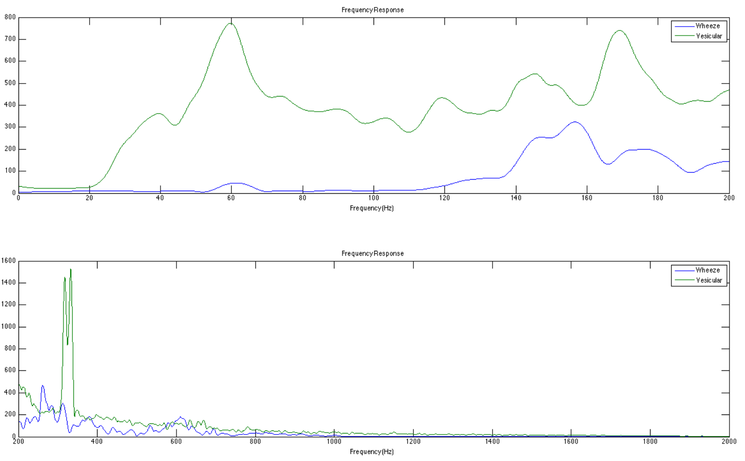

Figure 3: Comparison of vesicular and high pitched wheeze in frequency domain

The following features are extracted to characterize wheeze, with the motivations behind choosing each feature:

- Peak frequency in low frequency region (<200 Hz)
  Capture magnitude of spike of low frequency wheeze

- Peak frequency in middle frequency region (200-800 Hz)
  Capture magnitude of spike of high frequency wheeze

- Peak frequency in high frequency region (>800 Hz)
  Capture magnitude of spike of high frequency wheeze

- Ratio of max signal amplitude in low frequency region to average signal amplitude in low frequency region
  Capture a spike of low wheeze, which should have a high maximum to

average magnitude ratio, and normalizes across different signals

- Ratio of max signal amplitude in middle frequency region to average signal amplitude in middle frequency region
  Capture a spike of high wheeze, which should have a high maximum to average magnitude ratio, and normalizes across different signals

- Ratio of signal amplitudes across low frequency region to middle frequency region
  (Explanation)

- Ratio of signal amplitudes across high frequency region to middle frequency region
  (Explanation)

4.2.2 Crackle

The following features are extracted to characterize crackle, with the motivations behind choosing each feature:

- Ratio of signal amplitudes across low frequency region to middle frequency region
  (Explanation)

- Ratio of signal amplitudes across high frequency region to middle frequency region
  (Explanation)

References

[1] Yi, Gina A., and John V. Guttag. A Software Toolkit for Acoustic Respiratory Analysis. Thesis. Massachusetts Institute of Technology, 2004. Cambridge: Massachusetts Institute of Technology Libraries, 2005. DSpace@MIT. Massachusetts Institute of Technology. Dept. of Electrical Engineering and Computer Science. Web. 01 Apr. 2014.

[2] "Lecture 10: Speech Signal Analysis." Introduction to Computer Programming with MATLAB. UCL Department of Phonetics and Linguistics, n.d. Web. 01 Apr. 2014.