The Development of JEEM: Musical Instrument Recognition
A CS51 Final Project: Final Submission

April 30, 2015

Video Summary: https://www.youtube.com/watch?v=dAiDpJAz_xI

Team Members:

Erik Godard - erikgodard@college.harvard.edu
Mandela Patrick - mandelapatrick@college.harvard.edu
Elana Simon - elanasimon@college.harvard.edu
Joshua Meier - jmeier@college.harvard.edu

Dependencies:
See github repository for dependencies and setup instructions:
https://github.com/joshim5/CS51-FinalProject

Project Evolution:
Since submitting our original proposal, our project evolved significantly. At first, we were not well informed regarding the use and development of machine learning algorithms. We proposed using neural networks, an unsupervised method of machine learning, which we later learned was inappropriate for this project. Since our project primarily focuses on classification, we shifted our approach to the use of supervised techniques. In particular, we implemented a Support Vector Machine, Random Forest Classifier, and Nearest Neighbor Classifier.

In terms of the relevant feature extractors, we first planned to use Yaafe, but instead used mfcc, since we could code in Python.

Since we were able to correct this fairly easy (with our project TF's suggestions), we would say that our original planning was quite good. This allowed us to focus on the coding and fulfill every one of our milestones.

Milestones: We fulfilled every one of the proposed milestones. Our final specification is attached, for reference. The only major change is the way we implemented functions -- we switched to an object-oriented framework as the project moved forward.

Design Experience: As we furthered our development, we gained a better appreciation for the design of object-oriented and modular oriented frameworks. For the machine learning portions, we designed a mini library for our algorithms. We included a superclass for the machine learning algorithms. Each subclass implements a different algorithm. Therefore, all reused code is abstracted away.

Surprises: In order to divide work on the project, we split up into a machine learning team and a feature extraction team. We would meet together to make sure our code would interface nicely, but this team structure allowed us to better focus on our specific roles. Interestingly, the machine learning team had very high accuracy when testing the algorithms -- probably because the test samples we had gathered played a single note, did not have much noise, etc. (the same conditions under which the training data was collected) -- however, when using real-world data by actually recording instruments and testing these sound files, our classifiers became much less accurate. We realized that all of the audio files we trained the program with were from the same set of instruments we were using to test it. Thus, the algorithms could've been trained to differentiate qualities of the specific instruments that were irrelevant to the actual type of instrument (ie. it might've trained the program to differentiate the instruments based on the sound levels or background noise) and when we tested it on files from the same original dataset, these traits were enough to classify correctly.

Problem solving: We initially averaged the results of the mfcc coefficients over time. However, we found that we were losing a lot of the data by doing this, which led to our algorithm overfitting and performing poorly on real data. Consequently, we opted to use the unaveraged data, which ended up being a 300x13 matrix. This did not significantly impact the running time of our algorithm.

Next steps: If we were to continue this project we would add more instruments to our training set so that it could recognize more than 4 instruments. We would also continue to find more recordings of the instruments we already have so that our classifier would be better trained and more accurate. Additionally, we would improve the interface to make this program more user-friendly.

Group members contributions: Mandela and Erik worked on extracting the features from the audio files while Josh and Elana set up the machine learning algorithms. Then, as a team we worked on integrating the two parts and fixing any issues that arose.

The Development of EMEJ: Algorithms for Musical Instrument Recognition
A CS51 Final Project: Final Specification

April 17, 2015

Team Members:

Erik Godard - erikgodard@college.harvard.edu
Mandela Patrick - mandelapatrick@college.harvard.edu
Elana Simon - elanasimon@college.harvard.edu
Joshua Meier - jmeier@college.harvard.edu

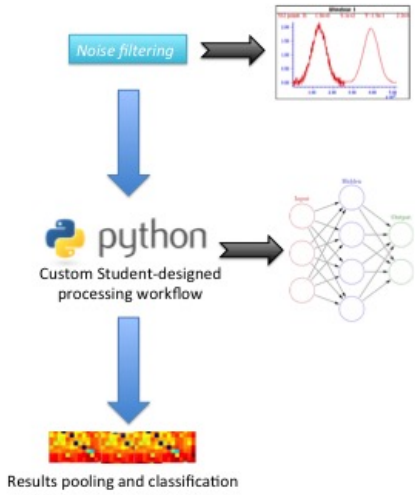
Brief Overview

The goal of this project is build a music recognition engine to determine the type of musical instrument being played in a piece of music. Given an audio recording of a music instrument, we will employ hand-crafted feature extractors and machine learning algorithms to categorize the type of musical instrument being played.

To accomplish these goals, we will train our machine learning algorithms using a variety of datasets obtained from the scientific literature. We will run a variety of statistical measures to determine the most appropriate dataset. We intend to use our project in an industrial setting and hope to choose the most appropriate dataset for this purpose.

After training our dataset, we will attempt classification using a variety of machine learning techniques based on random forest, support vector machines, and others. In particular, supervised learning approaches will be used in this project. Before analysis, we will divide our data into 80% for training (training set); 20% for testing (testing set). After training our algorithms using the training sets, we will experiment both on provided test data along with data collected from Harvard student musicians.

In addition to the pipeline described above, we may need to employ noise filtering for better results. Our workflow is summarized in the student-created schematic below. Please see the specific function descriptions that follow at the end of this specification.



In order to ensure the success of our project, we plan to employ agile and incremental computational practices. Each member of our team has written problem sets before, but we hope to model our development practices around a start-up model. Tasks will be designated; members will work independently; and “standup” meetings will be held during mini-team hackathons as we push major milestones on our project. Additionally, our group plans to employ *pair programming* with the goal of learning programming techniques from each other.

With these streamlined development practices, we are confident to complete the project and progress through stretch goals. With the large variety of technologies employed in this project, every team member will greatly expand his/her development skills. All development will be logged at https://github.com/joshim5/CS51-FinalProject.

Feature List

Core Features

- Front end that accepts audio input
- Algorithm that extracts features from the sound data
- Algorithm that classifies the instruments using machine learning

Cool Extensions

- Expand the number of instruments the app can recognize
- Recognize more than one instrument playing simultaneously

Technical Specification

Technologies for this project include:

- The Python Programing Language (http://www.python.org) -- Machine learning algorithms
  - `scikit-learn`: Machine learning in Python (http://scikit-learn.org/stable/)
- Git for version control
- iPython Notebook (http://ipython.org/notebook.html)
- Yaafe - audio features extraction (http://yaafe.sourceforge.net/)
  - While Yaafe was originally designed in C++, it supports Python bridging, making it convenient for the purposes of this project

Our project will be broken down into the following technologies:

- Front end - We will provide small functions that aid in the use of our project. While we will not be building an entire UI for the purposes of this project, we will still employ the use of “static strings” or command line interfaces (or both!) so that the user can test our program. We will cache the results from the machine learning algorithms to aid in development and testing (this can be done using iPython Notebook -- resource is included above)
- Feature extraction - This portion of the application will take the raw audio input and extract features so that we may classify the instruments. We do not know exactly which features we will use at this time, but possible candidates include pitch, frequency, volume, and many others. We will use the Yaafe library (http://yaafe.sourceforge.net/) to perform the feature extraction, and we will explore many of the capabilities that the library offers. As part of our stretch goal, we will investigate attempting to differentiate between two or more instruments being played simultaneously.
- Machine learning algorithm - We will train supervised learning algorithms to classify the instruments. There is similar work in the field (i.e. https://cs.nyu.edu/~gazzillo/machine_learning/results.html) so we know that the project is feasible. However, the NYU study used unsupervised techniques; our goal is to implement a similar project, but using supervised learning in Python. The members of our group have only a cursory knowledge of machine learning, so we will all learn the specifics of the algorithm. Depending on the time we have available to work, we may investigate writing our own algorithms or make use an off the shelf library such as http://scikit-learn.org/stable/.
- Output - Closely related to the front end, we will create a means of outputting the result of our algorithm. A list of instruments being played will be outputted to the command line or to a specified text file.

Next Steps

- Determine the data set we are going to use
- Secure access to instruments for testing
- Install the audio processing library and explore processing sounds

- Install the machine learning library and explore the implementation / use of supervised learning techniques

Timeline

- Segment 1
    - This week is denoted to setup.
    - Determine appropriate data sets and download/set them up
    - Secure access to instruments for testing
    - Setup machine learning libraries
    - Distribute roles for future coding
- Segment 2
    - In this week, we will focus on audio extraction
        - (i.e. Yaafe music extraction library)
    - The goal is to find specific feature vectors that can help determine the specific instrument
- Segment 3
    - Here, we will be performing the actual machine learning
    - Experiment with different learning techniques → decide on the best one
    - Using iPython Notebook, we can collect data and compare the approaches later
- Segment 4
    - Using the best learning technique:
        - implement the final pipeline
        - cache data so that the end-user can use the project quickly
    - Submission and complete write-up*

*Note: the final write-up will be completed throughout each stage of the project. For example, after the audio extraction implementation is complete, we will explain how it was conducted.

Further note: The work being done in each of these weeks can be done simultaneously (in particular, segments 2 and 3). This makes it easy to distribute work. Each segment corresponds approximately with one week.

See the specific functions at the end of this specification. In Week 1, our team will meet to discuss the distribution of these various tasks.

- Determine the data set we are going to use
- Secure access to instruments for testing
- Install the audio processing library and explore processing sounds
- Install the machine learning library and explore the implementation / use of supervised learning techniques

Resources

Similar Project:

https://cs.nyu.edu/~gazzillo/machine_learning/results.html

Music Sample Databases:
http://yaafe.sourceforge.net/
http://scikit-learn.org/stable/
https://cs.nyu.edu/~gazzillo/machine_learning/results.html
http://theremin.music.uiowa.edu/MIS.html
http://freesound.iua.upf.edu/index.php
http://www.findsounds.com/
http://dsp.stackexchange.com/questions/16994/feature-extraction-for-sound-classification

Progress Report

- Week -1
    - Draft specification
    - Decided on project/team
- Week 0
    - Met with TF
    - Wrote up specific functions
    - Wrote final specification

Functions

Since we are using Python, there is no formal interface. Instead, we have summarized the primary functions necessary for this project and grouped them by category. A brief description of what the function should perform is also included as bullet-points, to aid in our development and further outline what we will build.

Standard Functions

- initialize
    - Setup variables, initialize machine learning library, setup audio collecting, setup event handlers
- startRecording
    - Called when a button is clicked that denotes the starting of recording
    - Start recording
    - Setup a callback to endRecording that is called after a fixed amount (10 - 15 seconds) later
- endRecording
    - Called after a set amount of time (10 -15 seconds) has passed since the start of the recording
    - Stop recording and call extract audio, passing in the audio that was recorded
    - Do any cleanup associated with the recordings
- outputResult
    - takes as an argument the classification of a sound not in the training set and outputs its classification to the screen

Machine Learning Functions

- extractAudio
    - Takes as input the audio that was recorded
    - Runs feature extraction on the audio using the Yaafe library
    - Returns an associative array containing the features and their respective values
- extractTrainingData
    - Calls extract audio on each of the values in our training set
    - Maintains an array consisting of all of the returned values
- trainNeuralNetwork
    - Takes as input the results from calling extractTrainingData
    - Using scikit-learn, trains a neural network using the extracted features
    - Outputs the results so that they can be stored in a variable or saved for later use in additional instances of the program
- recognizeSound
    - Takes as input the features extracted from the sounds not in our training set

- Using the neural network that was previously trained, this functions runs the neural network with these features as input.
- The neural network will then return a classification, which this function will return

Progress Report

This week, we met to evaluate the work we had to do and assign tasks to each person. We determined that Erik and Mandela would work together on assembling the data set and extracting features and Joshua and Elana would work on implementing the algorithm. Each group's progress write-up is given below. In summary, Erik and Mandela successfully extracted features from audio, and Joshua and Elana successfully prepared a machine learning framework. In week 2 of development, we plan to integrate these components, resulting in the final music recognition software.

Erik & Mandela

We searched online for samples from which to build our training and test sets from, and we found a database from the University of Iowa that offered high quality recordings of instruments (http://theremin.music.uiowa.edu/MIS.html). We downloaded roughly thirty audio clips for each of the instruments we are planning on testing. Each sample features one of the instruments playing a note for roughly three seconds.

The next task was to convert the sounds files to the correct format. The progam we used to process the audio only accepted .wav files, so we used ffmpeg (http://ffmpeg.org/download.html) to convert each of the audio files, which were originally in .aif format, into .wav files.

Next we researched the best way to extract features, and we found a property known as Mel-Frequency Cepstrum Coefficients (http://en.wikipedia.org/wiki/Mel-frequency_cepstrum). They are frequently used in speech recognition and audio processing, and they provide an indicator of the timbre of the sound. Conveniently, we also found a python library that reads in a .wav file and outputs a matrix corresponding to the MFCCs of the sample (http://python-speech-features.readthedocs.org/en/latest/). The matrix contains 13 columns and rows proportional to the length of the audio, typically one per 10ms of audio.

We downloaded and installed this library, and then we loaded each of the sound files and used the program to extract the MFCCs for each files. We are still determining how to best utilize the coefficients in our machine learning algorithm. We think we can either average the values in each column or feed in the entire matrix. For now, our program outputs the MFCCs for each file into a json file, which the machine learning portion of the program can then use.

Joshua & Elana

We researched specific machine learning techniques to determine which would be most appropriate for this project. First, we determined the appropriate libraries, deciding that Scikit-learn would be best suited for the type of work we were conducting. In particular, the thorough Skikit-learn documentation was very helpful for our continued work.

Next, we decided upon the appropriate algorithms. Since we planned to write code that would distinguish between instruments, and since the data sets we would be using were already labeled (see Erik & Mandela's part above), we decided to use a supervised learning algorithm known as the SVM. Using feature vectors extracted from the audio data, an SVM represents the data in multi-dimensional space and determines a hyperplane which decides between the data. This gives binary results, but that works well for our project — the algorithm works separately for each instrument. We get a binary reading for each instrument — we return to the user the instrument for which the SVM returned "yes".

There is, of course, the possibility that the SVM will return multiple values. We will investigate a solution to this case in week 2 of the project.

Before integrating the feature vectors extracted from Erik & Mandela's component of the project, we designed and built our SVM using dummy data. In particular, we gave the machine test data in the format (x,x) -> x. For example, (3,3) -> 3 and (1,1) -> 1. Essentially, we are extracting the x-component = y-component of a point on the line y=x. We wanted our machine to find the closest point on the line x=y.

Scikit-learn examples involving SVMs were particularly important for this development. In particular, we began our project inspired by facial recognition algorithms, since the algorithm problem is very similar to music recognition.

In week 2, we will work toward advancing our SVM to identify music.