Speech Recognition using MATLAB

Challenge 4

Pablo Martin Garcia
ECE 160 Multimedia Systems
UCSB
Santa Barbara, CA
martingarcia@umail.ucsb.edu

Antonio Javier Samaniego Jurado
ECE 160 Multimedia Systems
UCSB
Santa Barbara, CA
samaniegojurado@umail.ucsb.edu

I. Introduction

In this project we will write a program to recognize spoken digits, in particular digits that go from 0 to 9. In order to do, we have to train the machine with a dataset. This dataset consists of 3 speakers with 50 recordings of each digit per speaker, so 1500 total recordings, which will be read from a directory called "audio". The algorithm we have created to train the machine and get the trained data is additional and complimentary to the algorithm that we are submitting to recognize the spoken digits that we later feed the machine.

To train the data we have used a built-in application from MATLAB after having extracted all the features that we consider the most valuable to recognize speech. After this, the training data is saved and then loaded into the new speech recognition algorithm that is the one that is going to be uploaded.

II. Extracting Features

A. Read Audio Files

Firstly, the algorithm we created to extract the features that we later use to train the data has to read all the files from a directory "audio". It will read all the speech files and saved them in a cell with all of them, sorting them by 0 to 9. This cell will be used also to save the features that will be extracted from each of the speech recordings. The function we use to read the audio files is *audioread* which gives us a vector of the audio signal as well as the sample frequency of it. We have noticed that some of the signals are in stereo, and so we have reduced them to use in mono for practical purposes.

The cell called *C_recordings* will follow the following structure: C_recordings{x,y,z}, where x corresponds to the digit in particular (which indexes go from 1 to 10, in other words, from 0 to 9), y corresponds to the different recordings (50 per speaker, 150 in total), and z are the different parameters. The different parameters are: 1, belongs to the audio_signal, or the vector with the values of the speech; 2, autoregressive power spectral density estimate in dB; and 3, the frequency vector, f, in cycles per unit time of the autoregressive power spectral density.

B. Perform the AR PDS

The Autoregressive Power Density Function (AR PDS) estimates the spectral density of a random signal from a sequence of time samples of the signal using the Yule-Walker method. Intuitively, it characterizes the frequency content of the signal. The autoregressive model specifies that the output variable depends linearly on its own previous values and on a stochastic term (imperfectly predictable term), and thus the model is in the form of a stochastic difference equation.

This is calculated with the function *pyulear* from MATLAB. It takes the input signal or audio recording, the order (the order of the autoregressive model used to produce the PDS estimate) and the amount of points in the discrete Fourier transform that are going to be used for the operation, denoted as nfft, and then it outputs the AR PDS as well as the frequencies of each of the values.

With this representation we get different AR PDS for each of the digits, which we can use to recognize on digit or another. To illustrate this better, we plotted all of the AR PDS of every digit recorded by one speaker in decibels. This plot is shown below in Fig. 1.

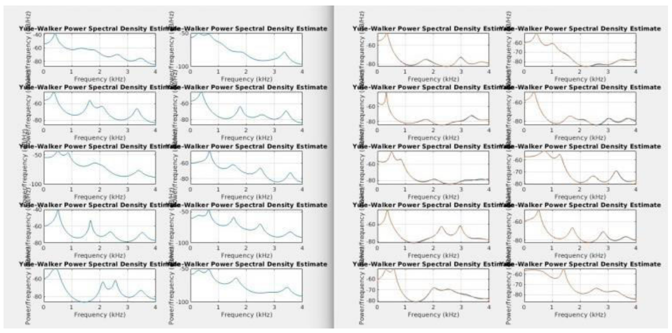


Fig. 1. Subplots of the ten PDS of each different digit from two different speakers, one in blue and one in red. The PDS are plotted in dB and follow the Yule-Walker method.

As you can see, the plot shows different frequency contents for each digit. We will then use different parameters of the AR PDS to train the data, such as the different frequency peak values of the AR PDS.

C. Calculate AR PDS peaks

In order to have more information of the PDS available to us and have more accuracy when recognizing each digit, we are going to extract the peaks of the AR PDS to give us a broader sense of what the spectral density of the signal looks like. To find the peaks, we use the built-in function in MATLAB called *findpeaks* which just like the function *max* will give the value of the peaks as well as the positions or frequencies where the peaks are. We will be more interested in where the peaks are and not the actual values of them because these can vary in amplitude.

However, to know what peaks are more important and more predominant, we need to take a look to the relative amplitude in the AR PDS so as to know which ones are higher. This is why both actual values of the peaks and the positions of them are stored in a matrix and then sorted by values of amplitude with the built-in function *sortrows* in MATLAB. Then with this new sorted matrix we will only extract from it the positions with the highest values, in other words the frequencies with highest amplitudes. This feature will give a broader sense of what the PDS looks like.

D. Calculating the Spectral Centroid

The reason why this parameter is useful is because it is a result of evaluating the center of mass of frequencies making use of the Fourier Transform, in other words, analyzing the frequency domain. Calculating the spectral centroid, we are finding the average frequency weighted by amplitudes, divided by the sum of amplitudes.

The spectral centroid of X is given by the following equation (Eq. 1):

$$Spectral\ Centroid = \frac{\sum_{k=1}^{N} k \times X[k]}{\sum_{k=1}^{N} X[k]}$$

(Eq. 1)

where k is the sample of X in the frequency domain.

The spectral centroid will be stored in a vector of length 1500 and this will be used as another feature to extract the trained data.

E. Calculating the Zero-Crossing Rate

The Zero-Crossing is a parameter that counts how many time the signal crosses the zero value, either from positive to negative or from negative to positive, as the name says. To clarify, to calculate this parameter the algorithm should count the number of times the amplitude of the signal changes sign.

The parameter is defined by the following equation (Eq. 2):

$$ZCR = \frac{1}{T-1}\sum_{t=1}^{T-1} func\{s_t s_{t-1} < 0\}$$

(Eq. 2)

where s is the sound signal of length T measured in time and the function inside the summation equals 1 if the comparison is true and 0 otherwise.

This feature will be stored as well in a vector of length 1500, one for each audio signal, and will be used as a feature to extract the training data.

III. Results

A. Training the Data with a Classification Learner

To train the data we use a MATLAB application called "Classification Learner". What this application does is it takes a matrix with certain values and you can choose if it should take the features from its rows or its columns. Then it will run an algorithm that will classify the data with different models and give a ratio of accuracy of the learned data given a specific model.

The matrix that we will be feeding the classification learner follows a certain structure. We store the actual digits in the first column of the matrix. This is what we will tell the classification learner to predict. After this, the following columns are parameters of each one of the recordings corresponding to every digit, this being the frequencies of the peaks of the AR PDS, the spectral centroid and the zero-crossing rate. Also, we will import this matrix from the MATLAB Workspace. You can see the classification learner application with the data fed into it before training it in the image below (Fig. 2):

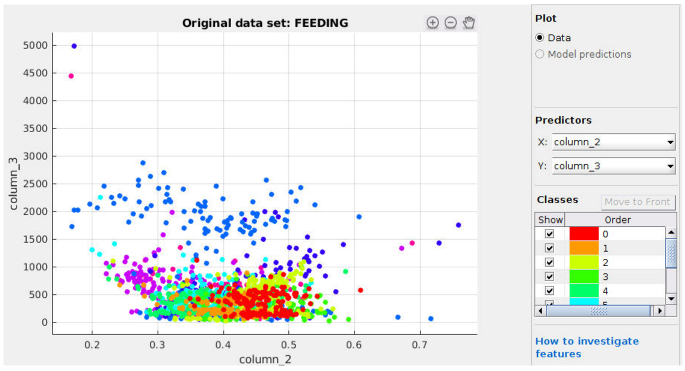
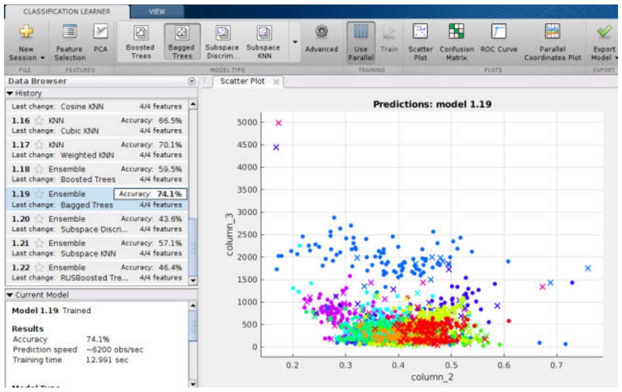


Fig. 2. Original data set, x axis is the zero-crossing rate and y axis is the spectral centroid. Each color represents a different digit.

Then we would specify that our classification learner goes through all the model types to see which one has highest accuracy. After running the algorithm and training the data we get that the highest accurate model is actually the Bagged Tree Model, with an accuracy of 74.1%. You can see the accuracy in the image below (Fig. 3):



History

		Last change: Cosine KNN	4/4 features
1.16	KNN	Accuracy: 66.5%	
		Last change: Cubic KNN	4/4 features
1.17	KNN	Accuracy: 70.1%	
		Last change: Weighted KNN	4/4 features
1.18	Ensemble	Accuracy: 59.5%	
		Last change: Boosted Trees	4/4 features
1.19	Ensemble	Accuracy: 74.1%	
		Last change: Bagged Trees	4/4 features
1.20	Ensemble	Accuracy: 43.6%	
		Last change: Subspace Discri...	4/4 features
1.21	Ensemble	Accuracy: 57.1%	
		Last change: Subspace KNN	4/4 features
1.22	Ensemble	Accuracy: 46.4%	
		Last change: RUSBoosted Tre...	4/4 features

Current Model

Model 1.19: Trained

Results

Accuracy	74.1%
Prediction speed	~6200 obs/sec
Training time	12.991 sec

Fig. 3. Data trained using prediction 1.19, or Bagged Trees, showing an accuracy ratio of 74.1%.

As you can see, our trained data classifies with a high level of accuracy the new data coming in and it recognizes with a certain level of certainty the digits, considering that a random guess would have an accuracy of 10% (one out of ten digits).

Also, in the plot you can see dots and crosses. Dots are correct predictions using that model, while crosses mean an incorrect prediction using that model.

After this, we are going to plot the training data in respect of different features so that the different regions of each digit are clear. The plots are below (Fig 4 to Fig 9).

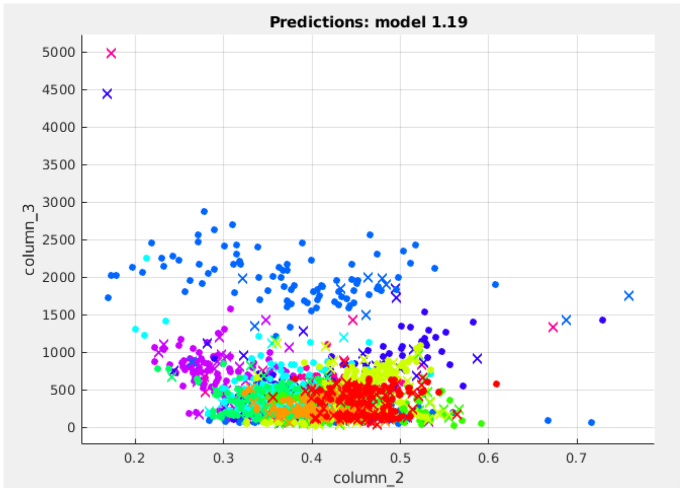


Fig. 4. Trained data set, x axis is the spectral centroid and y axis is the zero crossing rate. Each color represents a different digit.

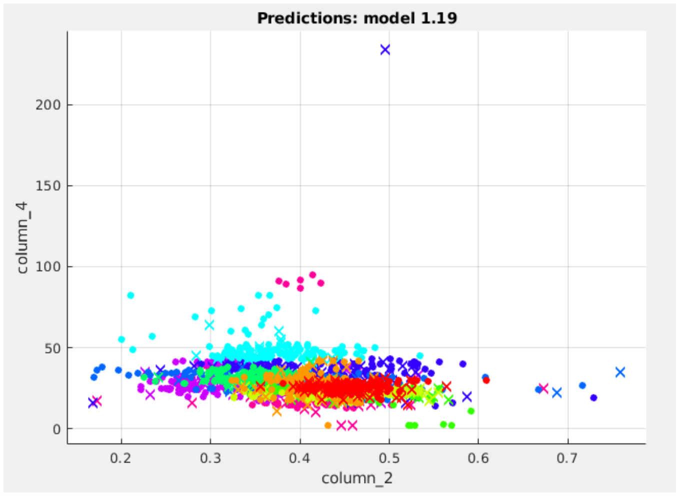


Fig. 5. Trained data set, x axis is the spectral centroid and y axis is the first peak. Each color represents a different digit.

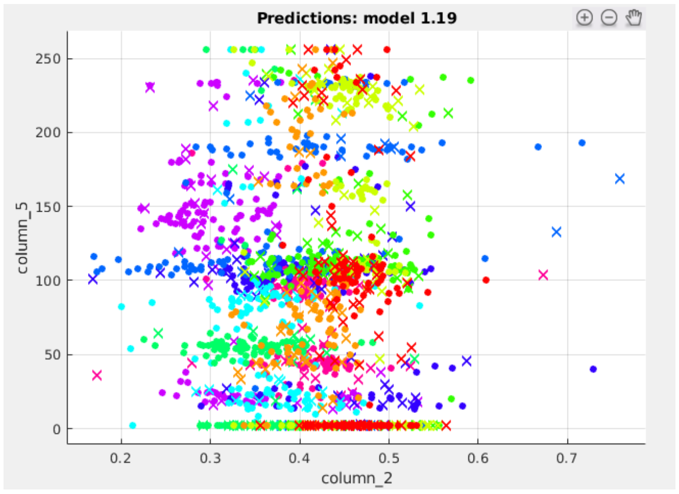


Fig. 6. Trained data set, x axis is the spectral centroid and y axis is the second peak. Each color represents a different digit.

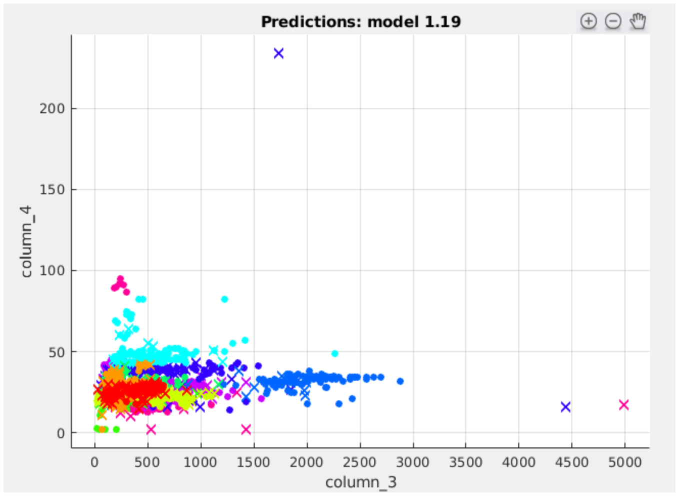


Fig. 7. Trained data set, x axis is the zero-crossing rate and y axis is the first peak. Each color represents a different digit.

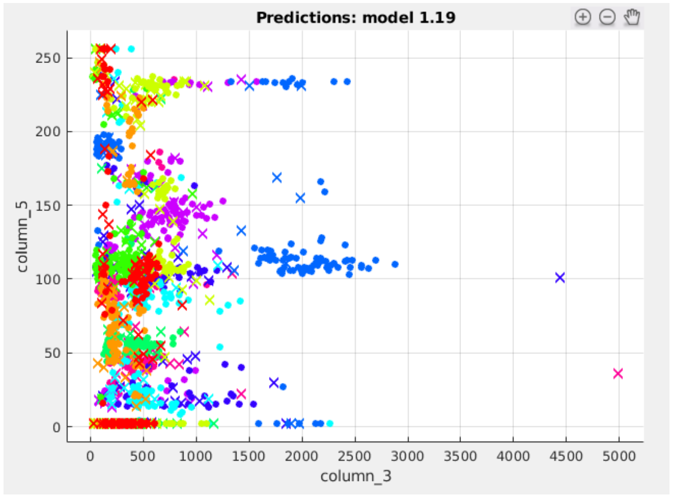


Fig. 8. Trained data set, x axis is the zero-crossing rate and y axis is the second peak. Each color represents a different digit.

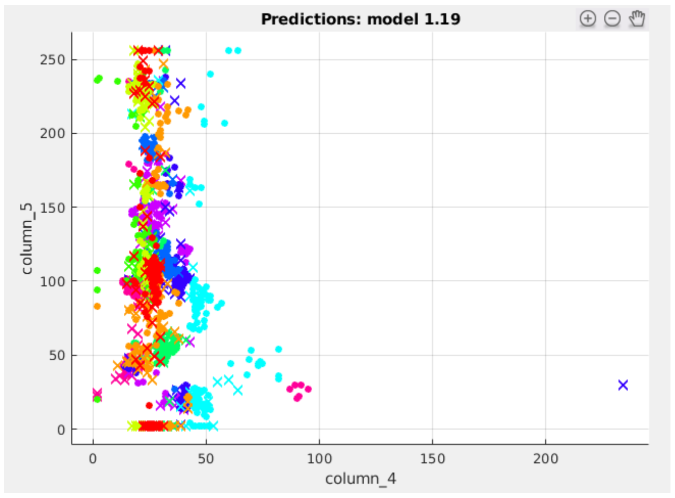


Fig. 9. Trained data set, x axis is the first peak and y axis is the second peak. Each color represents a different digit.

As you can see, the digits are not that well divided especially in the last features, but by having multiple features the accuracy of the classification is high.

IV. Conclusion

Speech recognition is one of the most useful areas of applications and research nowadays. Our algorithm shows a data-driven way to process audio signals and recognize spoken digits from 0-9 with a pretty decent accuracy (74%).

To achieve this goal, the process of feature extraction and data training results in a very effective way to later predict new audio. In our case, we have seen how looking at the PSD, Zero Crossing Rate and Spectral Centroid we can successfully distinguish speech properties. After training our data based on such features, the algorithm effectively extracts those from the input audio to be tested and predicts which digit it fits best with by using the Bagged Tree prediction method. We have designed it to use that method as it showed the highest accuracy.

So far this has been the most useful project we have worked on, given its impact on technology in today's industry, so in that sense we also feel fulfilled.

.