ENGR 518 PROJECT REPORT

School of Engineering
Faculty of Applied Science
University of British Columbia

Project Title: Speaker recognition
Group No.: 14
Members: Juliya Johnson, Shubham Mohapatra, Kaimeng Du
Date: 2023 Nov 29

Introduction

Speech recognition is widely used in current smart devices, including phones, smart speakers, as well as home controllers. Since the voice pattern of each user is different, identifying each speaker can improve the accuracy of voice recognition. Moreover, in recent Google and Siri smart speakers, smart assistance can access personal calendars, notes, contacts and reminders based on the detected speakers. Speech recognition technology can utilize the information from speaker's voice spectrum, talking speed, punctuations or even frequently used words to make a comprehensive identification. Yet, in this project we explored the mechanism of speaker recognition with a simple 3-class classifier only based on speaker's voice spectrum with logistic regression.

Theory

In this project we will build a 3-class classifier based on logistic regression. Sigmoid will be used for activation function and cross-entropy cost function will be utilized for training. Data will be collected, processed, labelled, shuffled and split into training, validation and test datasets. A classifier will be trained with training and validation data set and tested with testing data set.

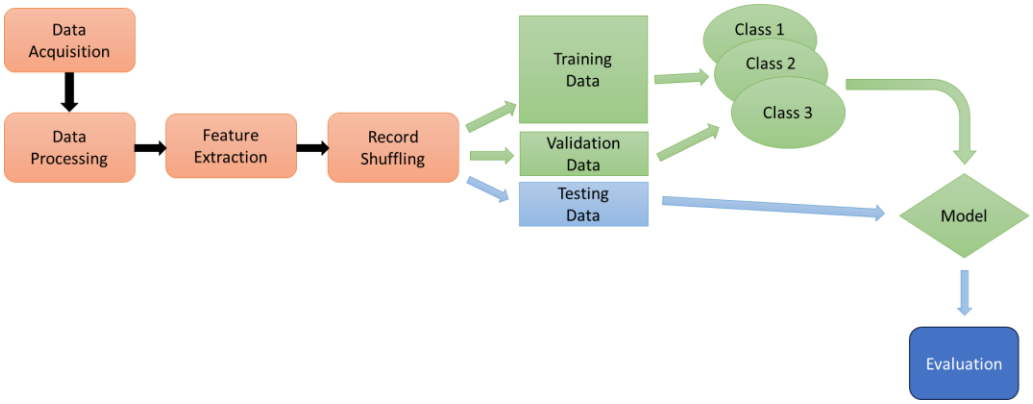

Figure 1 Design flow of our 3-class classifier

Algorithm

Data Collection

3 subjects were recruited in this test. From each subject, around 10-minute book-reading voice data was recorded with numerous voice recorders with different noise backgrounds and the best recorder (R3312, Aigo) was chosen using 16kHz sampling rate and 16-bit data width in wav format which contained minimal noise levels.

Data Processing and Feature Extraction

The 2-channel wav files were read with scipy.io.wavfile.read(). Right channel was discarded. Short Time Fourier Transformation was conducted on a 2048-point moving window with 512 points overlap. Energy of these Fourier Transform results was calculated and used as features for later classification. By observation, the frequencies beyond 1kHz shows limited variations between subjects. So we only used frequencies below 1kHz for later training and testing, which equivalent to first ~150 points in the Fourier Transform results. To make the dataset size the same from each subject, only the first 1000 samples from each subject were used. Therefore, the total dataset contains 3000 records, where each record contains 150 features.

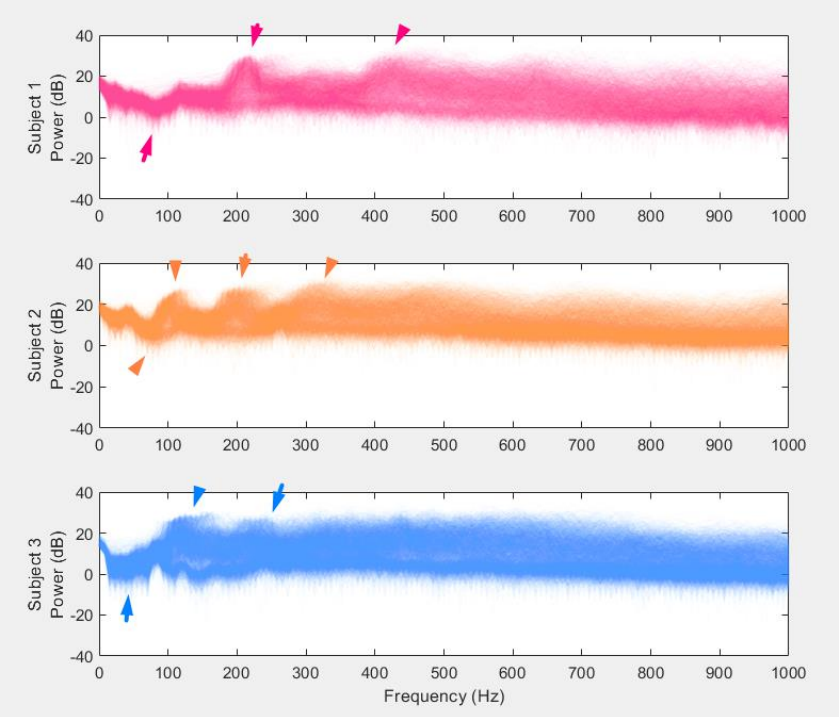

Figure 2 The Short Time Fourier Transform result from 3 subjects. The fundamental, first and second harmonics of the voice data from Subject 1 and 2 can be clearly observed (pink and orange arrows), while only the fundamental and first harmonic can be observed for Subject 3, the 2nd harmonic is blurry (blue arrows).

Data Labelling, Shuffling and Splitting

The data sets from 3 subjects were then stored in panda dataframe and labelled before they were shuffled. The data might be uniformly arranged and can belong to only one subject before shuffling. Due to this the model predicts all the data points to belong to a single class. To avoid this, the data points are randomly shuffled to make sure the model is trained well, and the

predictions are accurate. This means the order of the data points is randomized. The purpose is to ensure that any patterns related to the order of data do not influence the learning process, and that the model remains generalizable and doesn't learn any sequence bias.
After shuffling, 70% of the data is under the training set. The model mainly learns from this data subset, and it tries to learn patterns to get the desired results. 15% of the data is considered for validation for tuning and avoiding over fitting. The testing set also comprises of 15% for which the model is tested for new sets of data samples never encountered by the model yet.

Classifier Design

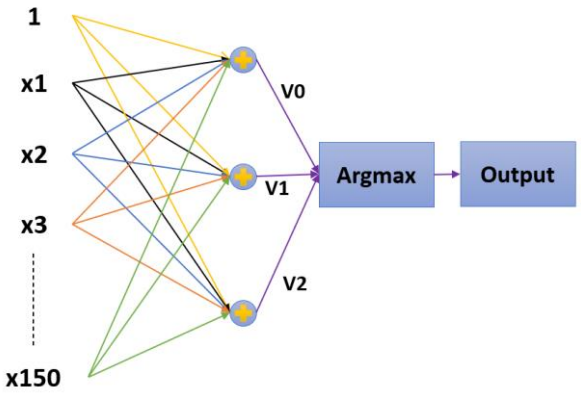

Figure 3 The structure for our logistic regression based 3-class classifier.

Our classifier took 150 input features, weighted sum them. The max of summation class label will be used as classified result. Therefore, the cross-entropy cost function for our model is:

$$g(w) = -\frac{1}{P}\sum_{p=1}^{P}\left[y_p \log\left(\sigma\left(\bar{x}_p^T \bar{w}\right)\right) + \left(1 - y_p\right)\log\left(1 - \sigma\left(\bar{x}_p^T \bar{w}\right)\right)\right] + \lambda \|w\|_2^2$$

where $\sigma()$ is sigmoid function; y_p is the class label {0,1}; $\bar{x}_p$ is the features with 1 appended on top of the vector; $\boldsymbol{w}$ is the feature touching weights; $\bar{w}$ is the feature touching weights with bias appended on top. P=3000 for our case.

This function computes the regularized cost (loss) using the cross-entropy loss function. It is used to train the logistic regression model by punishing the wrong classification result. The regularization term, controlled by λ, helps prevent overfitting by penalizing large weights.

Since logistic regression is inherently binary, this function extends it to handle multi-class problems using the one-vs-all (OvA) approach. It trained separate logistic regression weight sets for each class against the rest.

Results

Model Accuracy

The classifier was trained with 3000 iterations and the step size is 0.01. The accuracy was calculated by evaluating the correct classified record rate in the testing data set. Because the data shuffling and splitting was random, the performance of this classifier varied. But each time the accuracy achieved was around 95%. Because the 2048 moving window we used only

covered 1/8 second on the 16kHz sampling rate, there were plenty of Fourier Transform windows where no speech was covered. Therefore, we consider the 95% accuracy is good.

We used the same data set and fed it into a single layer neural network of Neural Net Pattern Recognition Toolbox in MATLAB. We tried 10, 100 and 150 neurons in hidden layers; and under each setting the neural network model was trained and tested 3 times. The Neural Network classification accuracy results were all around 95%. We believe our model has achieved the best performance given the nature of the data set.

Training Results

Training start time: 29-Nov-2023 17:53:30

Layer size: 150

	Observations	Cross-entropy	Error
Training	6300	0.1681	0.4967
Validation	1350	0.1786	0.5111
Test	1350	0.1706	0.5044

Figure 4 MATLAB Neural Net Pattern Recognition Toolbox training result of a single layer neural network with 150 hidden layer neurons. The accuracy is comparable with our logistic regression classifier.

Boosting

Signal boosting approaches in machine learning have been gaining ground recently. The key idea behind boosting is to emphasize the learning of difficult or misclassified examples by assigning them higher weights during subsequent iterations of model training. This iterative process focuses on improving the model's performance by correcting errors made by earlier models in the sequence. To perform boosting, repeatedly train models by decreasing regularized lambda parameter values to improve performance. When the lambda is reduced from a larger value to smaller value, it resulted in the test accuracy rate approximately from 33% to 90% as shown in below given fig.

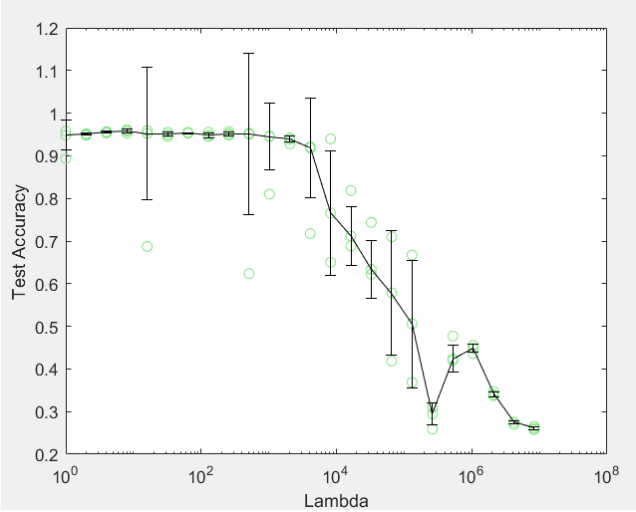

Figure 5 Model accuracy increasing can be observed with the decreasing of regularization parameters (lambda). The green circles present accuracy of one independent training/testing result. The error bar is presented as median ± std, because mean value can be pulled by outliers.

After training several weak models, boosting combines their predictions through a weighted majority vote or weighted averaging to create a more robust and accurate final prediction. Since,

the outliers significantly impacts the mean, median of weights are used to mitigate the influence of outliers. Figure 5 represents the median of weights obtained from boosting, which helped in more stable and robust prediction.

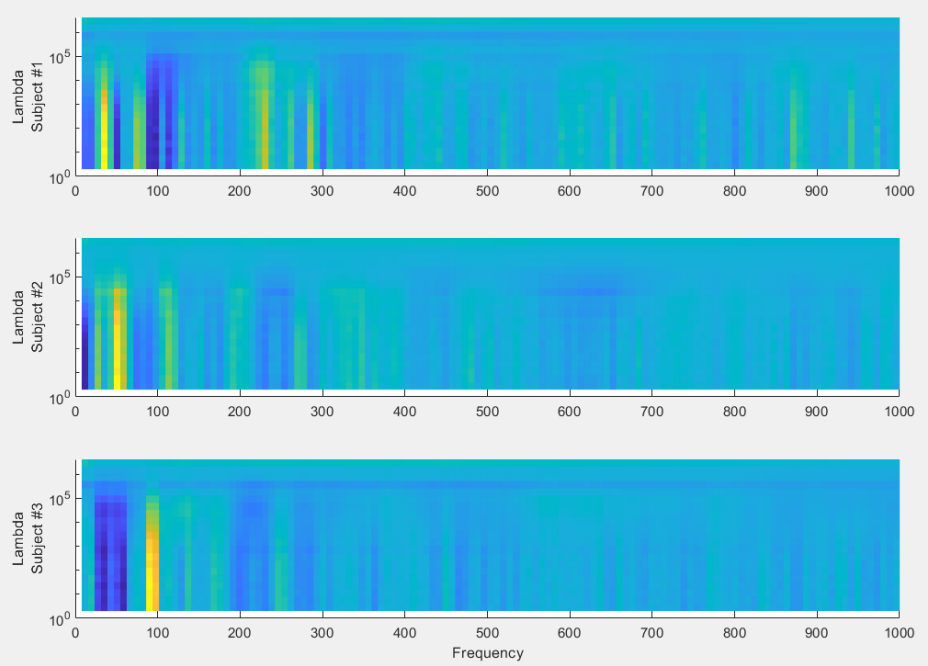

Figure 6 The heat map plot showing the relationship between trained weights of classifiers and the regularization parameter, lambda. When lambda value is large, the weights was penalized and therefore small. The weights begin to show and keep growing as decreasing of lambda, and plateaued when the accuracy was around 95%.

Feature Selection by Boosting

We select 6 frequency bands (7.8 Hz, 31.3 Hz, 46.9 Hz, 54.7 Hz, 85.9 Hz, 93.8 Hz, 109.4Hz) by observing the colours of Figure 6 where the blue or yellow colour were the strongest, indicating a larger value for classifier weight. Then we use only these 6 features to train our classifier. The result accuracy was around 82%. We were shocked that with only 6 frequencies, the accuracy above 80% is unexpected.

Conclusion

To summarize, in this project, we designed, coded, trained and tested a 3-class classifier based on logistic regression. The performance of this classifier is compatible with single layer neural network where hidden layer neuron number equals the input layer at around 95% accuracy.

We further explored boosting method for feature selection. We obtained 6 frequency bands where the weights are highest and trained another simplified classified based on only these 6 features. The accuracy was around 83%, which was acceptable.

Finally, we ported our model to a real time classifier, which will be used in our presentation. The difference between the model mentioned in this report and the model used for our presentation demo is covered in the appendix.

Appendix

For the model mentioned in the report, we used voice recordings from a voice recorder, which provided excellent noise suppression and frequency response. However, that cannot be used for our demo during presentation. So, we tried several other microphones, including Logitech USB H390 and LG HBS-780. We found cheap wired microphones suffer from great background noise. Therefore, we finally used LG HBS-780 for demo use. And since LG HBS-780 has a different frequency response with previous recorder, we recorded voices from subjects again and trained classifier for demo use.

The demo script also uses the first 150 points from STFT, but uses softmax function to show the probability of prediction.

The UI of demo script was built with wxFormBuilder.

Reference

Access Microphone with pyAudio by **Dataquest**

https://github.com/dataquestio/project-walkthroughs/tree/master/microphone