UCLA
Dept. of Electrical and Computer Engineering
ECE 114, Fall 2019
Speech Machine Learning Computer Assignment:
Native and Non-Native English Speaker Classification

1 Introduction

Machine Learning is an important tool in speech processing. It allows us to find relationships between measurable features of speech data and complicated characteristics that we wish to identify. In machine learning, we extract features from the raw data which is labeled as having or not having some characteristic. The features and the labels are then passed as inputs to a neural network. It is the neural network's job to approximate a function that can accurately map a given feature vector to the appropriate label. In this assignment, you will carry out every step of this process using state of the art software.

2 Data

The quality of the dataset used is very important in training a neural network. In this assignment, we will use a subset of The Speech Accent Archive from George Mason University. This is an extensive archive that includes recordings from thousands of speakers all reading the same passage of approximately 20-40 seconds in duration in English. The native language of the speaker is included in the filename (for example, the speaker in the file english1.mp3 is a native English speaker, the speaker in the file french1.mp3 is a native French speaker, etc). We will use a subset of 170 recordings (80 from native English speakers, 90 from non-native english speakers).

In machine learning, we often divide the data into 3 sets: The training set, the testing set, and the validation set. We might choose to use 80% of the data for training, 10% for testing, and 10% for validation. When we train the neural network, we will train it only on the training set. However, we do wish to avoid finding a solution that is too specific to the training set, so we

perform intermediate tests on the validation set. This gives us a measure of how well our network does on data that it has not seen before. If the training set accuracy of our neural network is high but the validation set accuracy is low, that means that we have found a solution that overfits to trends in the training set and need to add regularization to our network to find a more general solution. Once we believe that we have perfected our network, we can obtain a final evaluation of the system by testing it on our testing set. We should not use the testing set at any point in training.

3 Feature Extraction

We need to extract useful information from our data to pass into our neural network. We have two useful tools in MATLAB to extract commonly used speech features from audio data: VoiceSauce and VoiceBox.

3.1 VoiceSause

VoiceSause is an application in MATLAB from the UCLA Phonetics Laboratory which extracts several useful features from input speech audio files. Documentation and download are available `http://www.phonetics.ucla.edu/voicesauce/`. The features extracted from VoiceSause are mainly concerned with identifying key traits of a particular speaker or speaking style and have been shown to yield high performance in speaker identification tasks. VoiceSause is run on a directory of .wav files, and the output is one .mat file per .wav file in the original directory where each new .mat file has the name and estimated parameters of its corresponding .wav file. The example script voicesause_output2featvec_example.m shows how to concatenate the outputs into a matrix containing the feature vector of each sample.

3.2 VoiceBox

VoiceBox is a MATLAB based feature extraction package from the Imperial College, London. Documentation and download are available at `http://www.ee.ic.ac.uk/hp/s`. This library contains many features usefel for speech recognition, speech coding, speech enhancement, and speech synthesis that you might find useful for

this task. The example script feat_from_voicebox_example.m one way to extract the features and concatenate them using VoiceBox.

3.3 Labels

We need to assign labels to the data so that the neural network can learn to classify data points based on the given criterion. This is a binary classification task (Decide native or non-native English speaker), so we will assign a label of 1 to each recording of a native English speaker and a label of 0 to each recording of a non-native English speaker. This is done in the example MATLAB scripts.

4 Neural Network

4.1 Implementation

We will use the TensorFlow library in Python to implement a Neural Network and train it on our feature set. TensorFlow by Google is one of two (the other being Pytorch by Facebook) leading open source machine learning softwares used in industry. Experience with TensorFlow and Pytorch is a valuable resume building skill which is why we will have you gain some familiarity with one. Keras is Python library that calls TensorFlow through simplified commands, so we will use Keras to implement our network. Two neural network examples in keras are provided for you in the python notebook, ECE114_Speech_ML_project_example.ipynb. One is a feedforward neural network and one is a recurrent neural network.

4.2 Feedforward Neural Network

A feedforward neural network is the earliest and simplest neural network architecture proposed. It is an expansion of the single-layer perceptron, ie. a multi-layer perceptron or a collection of fully connected layers. Each node will output an affine combination of its inputs, and the network will train to learn the optimal weights and bias of the affine combination that best map the input feature vector to the corresponding label. The term "fully connected"

or "dense" layer means that each node forms an affine combination from every output of the layer before it, as opposed to just some of the those elements.

4.3 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are a type of neural network designed to interpret time-dependent data. The network assumes that the input is a time series and learns not only learns a relationship between the features inputed, but trends in the time evolution of the data. This is useful for speech data since we can consider speech to be a time-varying waveform. In the example provided, we use an implementation of the RNN called an LSTM (Long-Short Term Memory). An LSTM layer of an RNN creates a state variable whose value changes as the data progresses in time. The phrase "Long-Short Term" refers to the fact that this state variable may depend on time sample going very far back or only on recent values depending on the optimal solution for the network. We might say that the state variable can remember the history of the input that will help it make a new prediction and then forget past values of the input that are no longer relevant later in time. While powerful, large RNNs are computationally expensive to train and often impractical for real-time on-device applications.

5 Instructions

In this assignment, you will program a neural network to decide whether a given speaker is a native or non-native English speaker given a 20-40 second utterance. You will:

1. Decide what features work best for this task and extract them to .mat files. You may use VoiceBox, VoiceSause, any opensource tool that you find, or your own code to do this. Save them as feat_vec.mat or feat_vec_rnn.mat. Be able to explain why you chose these features, what they represent, and how they were calculated.

2. Run a neural network using your features. Download the Python notebook ECE114_Speech_ML_project_example.ipynb to your google drive

and open it in Google Colab. Set the runtime to GPU. To load your feature data into the network, you will need to place the .mat files in your google drive and mount your google drive to google Colab. Create a folder in the top level directory of you google drive called ECE114F19_SpeechMLproj and upload your .mat features to it (you can name the folder anything as long as you change the directory name in the notebook). Then you should be able to run the cells of the notebook without issue. You should change the architecture or parameters of the network examples provided to improve the performance of the network. Keras documentation is a available online at `https://keras.io/`.

3. Write a report detailing your design decisions and why you believe that they were successful or unsuccessful. This is not a solved problem, and you are not expected to find a perfect solution. However, you should use what you have learned in lecture to make reasonable design choices for speech processing and explain them.