Voice Recognition

Seroussi Yishay [✡] Bismuth Samuel [❄] Shaag Yehonatan [✦]

Deep Learning

Abstract

Given a voice recording, by using deep learning, we are able to determine (With high percentages of certainty) from what country the accent of the speaker coming, and then, make assumption about the origin of the speaker.

To achieve such a project, we need a strong data-set built from the website provided by our lecturer and some downloaded audio from Youtube. Our data-set includes each one of the next languages : French, Hebrew, USSR and English (UK and USA).

Each recording of the website is the next text : " Please call Stella. Ask her to bring these things with her from the store: Six spoons of fresh snow peas, five thick slabs of blue cheese, and maybe a snack for her brother Bob. We also need a small plastic snake and a big toy frog for the kids. She can scoop these things into three red bags, and we will go meet her Wednesday at the train station. " read by a subject.

1 Introduction

Voice recognition is an important subject of research in the world of technology. Indeed, in the all day life, a lot of speech recognition tools are used, like Siri or Alexa.

One of the major challenge is to recognize a non native speaker. Accent detection or classification can help in a lot of topics. As an example, accent recognition is used by soldier in the toll to check if either the car driver is a native speaker or not.

In the context of this project, the classification is done by using audio from people with a panel of five different accents. Our approach use the mfcc conversion from the audio files, and adding a script in python implementing a softmax

[✡] Student of Computer Science (third year), Ariel University, Ariel 40700, Israel. Id : 305027948. Email: seroussi1@gmail.com

[❄] Student of Computer Science (third year), Ariel University, Ariel 40700, Israel. Id : 342533064. Email: samuelbismuth101@gmail.com

[✦] Student of Computer Science (third year), Ariel University, Ariel 40700, Israel. Id : 308357953. Email: yoshago@gmail.com

using also neural network, we attempt to classify the speakers. This project have been divide into 4 steps. The first step was about to build a data-set. The second step was the implementation of the softmax, as simple as possible. During the third step, and to improve the test accuracy, we added some fully connected hidden layer. The fourth and last step was about to implement a convolutional neural network (CNN).

2 Related Work and Required Background

One of the interesting published paper was speaking about the accent classification of non-native English speakers [1]. On this paper, the publisher attempts to classify the next languages: Tamil, Germany, Brazilian Portuguese, Hindi, and Spanish. To do this, he used a MFCC, Delta and FBank conversion to convert the audio into numbers. Then, his classification is based on a softmax implementation, using Multi Layer Perceptron, Recurrent Neural Network with Long Short Term Memory, and Convolutional Neural Network. As a result, the best accuracy reach 52% of good guessing.

Another interesting publication was about accent classification but this time for 23 languages.
This paper [3] accentuate the data-set on record of simple word instead of entire sentence. All the audio are then extract using MFCC and PLP (Perceptual Linear Prediction). The best accuracy of this work is about 51%, using 8 features.

About speech recognition (not only accent recognition), a lot of work have been done, and is actually used in the all day life. The power of all the best result is explained by a strong data-set.

3 Project Description

In this project our approach involves audio treatment and conversion to make the computer understand as best as possible the audio file.

To realize such a thing we convert the wav file (audio file) into mfcc matrix. From the book [2] we can read:
"The first step in any automatic speech recognition system is to extract features i.e. identify the components of the audio signal that are good for identifying the linguistic content and discarding all the other stuff which carries information like background noise, emotion etc.

The main point to understand about speech is that the sounds generated by a human are filtered by the shape of the vocal tract including tongue, teeth etc. This shape determines what sound comes out. If we can determine the shape accurately, this should give us an accurate representation of the phoneme being produced. The shape of the vocal tract manifests itself in the envelope of the short time power spectrum, and the job of MFCCs is to accurately represent this envelope."

3.1 Data-set

The data set of the website is as the next table: Adding to each recording, for each subject we got information, such as the next table 1.

birth place	st. laurent d'onay, france (map)
native language	french (fra)
other language(s)	spanish
age, sex	20, female
age of english onset	12
english learning method	academic
english residence	usa
length of english residence	0.4 years

Table 1: Biographical Data

Adding of course the audio.

And, of course the data is also composed of some Youtube video converted in WAV files. This work takes only the native language of the five regions (France, USSR, UK, USA and Israel) and the audio that we convert into a WAV file. Seventy percent of the data is used for the train, the rest of the data will be used in order to test our data-set.

An important note is about the audio: Of course the computer don't understand audio such as human does, so, we have to translate the audio into numbers understood by a computer. To do this, and by Lecturer's advice, we use Mel-frequency (mfcc) to handle the audio. To do this, all the audio were downloaded in WAV files, and we use the module "python speech features" to transform the audio file into a matrix of numbers understanding by the computer.

Then, the actual data set is composed of a CSV table (ready to be read by a python script and easily alterable) with two columns, the first is the region and the second is the link to the WAV file.

3.2 Work description

To recognize accent, using Tensorflow, this work implements a multinomial logistic regression (softmax). First, a python script recuperates the data from the csv, and convert it into numerous python object, such that each object includes data about the accent of the recorder, and three seconds of audio. The data is splitted into four arrays. Two arrays for the train: one with the data set and the second with the labels and same for the test. 70% of the data is used for the train. In the implementation of the softmax, a lot of parameters can be modified. Our work for now will be to find the best values for these parameters that will optimize the loss function and the test accuracy.

3.3 Multi-layer Perceptron

The classifier can have multiple hidden layers. In this project, we find that two hidden layers in general perform better. Of course, the layer are fully connected. For both layer, there are 256 neurons. For the activation function in the hidden layers, "relu" is chosen for its efficiency. We choose the "adam" solver (optimizer) because it is an efficient stochastic gradient descent. The regularization penalty and initial learning rate are tuned. This was a good solution to the over-fitting.

3.4 CNN

The implementation of the CNN had the next parameters. Two layers of convolution and maxpool. The first layer, includes 32 filters of size 5x5. And the maxpool's size was 2x2.The second layer includes 64 filters of size 32x5x5. And maxpool's size of 2x2. The input matrix was about 13x99. The idea was to take some part of the matrix and make the computer focused on those part which can make a better understanding about the matrix in general by knowing which filters are the most important.

4 Experiments Results

4.1 First Attempt

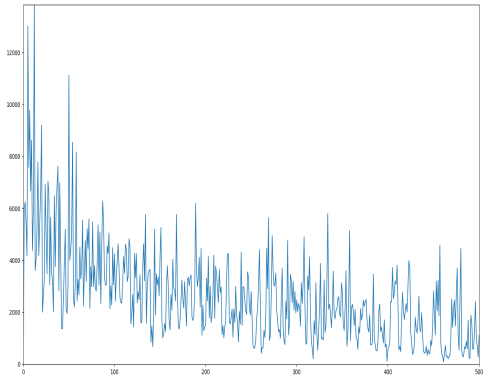
We made some tests trying to improve the results, such as the next table 4.1.

Test number	Data	Parameter	Remark
1	English,(UK and USA). USSR, French: 40 minutes of recording from 20 different people. Hebrew: 18 minutes of recording from 9 different recorders.	Gradient: 0.00001, features: 1, accuracy: 20%.	At this stage, the goal of the softmax wasn't understood yet. Note that we worked with a matrix of number converted by mfcc.
2	English,(UK and USA). USSR, French: 40 minutes of recording from 20 different people. Hebrew: 18 minutes of recording from 9 different recorders.	Gradient: 0.01, features = 3800, batch: 200, epochs: 5001, accuracy of the train: 100%, accuracy of the test: 35%.	We converted the matrix into an array to make the work easier. Also, we decided to divide each record into numerous records between 2 and 5 seconds.

3	French: 1 hour of recording from 30 people. Hebrew: 18 minutes of recording from 9 different recorders. Russian: 80 minutes of recording from 40 different recorders. UK: 55 minutes of recording from 36 different recorders, USA: 1 hour of recording from 33 people.	Gradient: 0.05, features = 12950, batch: 100, epochs: 500, accuracy of the train: 100%, accuracy of the test: 40%	We divide English into two classes: USA and UK. The conclusion of the result was an over-fitting, so we wanted to add regularization, sadly, without success. Same for normalization.
4	French: 2 hours of recording from 50 people. Hebrew: 38 minutes of recording from 14 different recorders. Russian: 2 hours 20 minutes of recording from 40 different recorders. UK: 90 minutes of recording from 29 different recorders, USA: 2 hour of recording from 50 people.	Gradient: 0.05, features = 6450, batch: 100, epochs: 500, accuracy of the train: 80%, accuracy of the test: 50%	We added more data from some other platform - YouTube. The new data is not necessary the "please call Stella" record. By adding the data we saw an improvement, we need to deal with the over-fitting, to be more exact.
5	French: 2 hours of recording from 50 people. Hebrew: 1 hour 30 minutes of recording from 28 different recorders. Russian: 2 hours 20 minutes of recording from 40 different recorders. UK: 90 minutes of recording from 29 different recorders, USA: 2 hour of recording from 50 people.	Gradient: 0.05, features = 6450, batch: 100, epochs: 500, accuracy of the train: 80%, accuracy of the test: 52%	We added more data more data in hebrew.

Table 2: Experiences

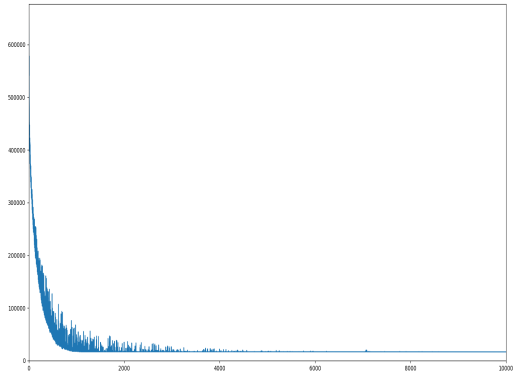
After a first submission, our test accuracy reached 50%. At this stage, no neural network or hidden layer was used. Please see the section work description which refer to all the experiences we made. Here are the picture of the loss function:



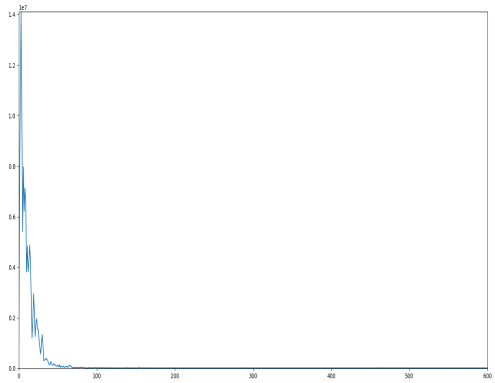
The conclusion after this first stage, was that the data was over-fitted. As the picture is showing the loss function jumps a lot and can't figure out the point which minimize the function. Another clue was about the train accuracy, which reach the 100% pretty fast, but the test accuracy was about 50% only.

4.2 Second Attempt

Adding two hidden layers to our code, we improve our accuracy of 5%:

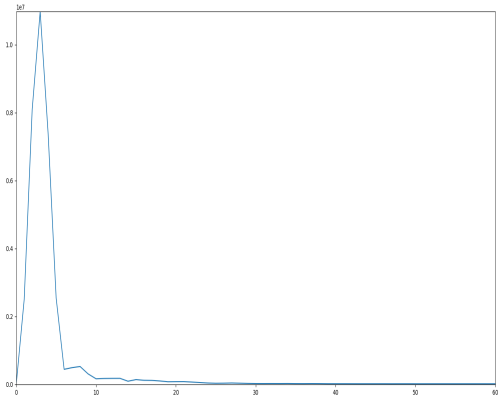


Notice that in particular on this picture, we try to change the epochs to 10000. To get the accuracy better, regularization helped, such that the accuracy have been improved once again by 5%, the loss function was like the next picture:



At this point it was important to approach it from a different angle:
(Failed approach)
Making a batch normalization was a good solution, so we looked up on the internet and found a code that use batch normalization. the code also used several different initialization to the weights such as normal, random, Xavier. we did not understand the meaning of the Xavier initialization and recalled that in the lecture we learned that the neurons weights should be initialized to random. we tried to adopt this code to our data with no success. at last, we decided to give up on this option.

Finally, by using the "relu" function for our hidden layer (we used add before), then, the accuracy improved to 77%, and here is a picture of the loss function:



Obviously, the use of the "relu" function from the module Tensor Flow make our accuracy much better, and also make our loss function nicer. This can be explained by the fact that the activation function is converting the linear space into a non linear space and making the system more complex.

As a short conclusion, we fix our over fitting by adding the "relu" function with hidden layers, and the regularization, and also, by adding data to our data-set.

4.3 Third Attempt

On this attempt the main goal was to add CNN to our softmax implementation. This attempt was a failure (28% for the test accuracy) for the next reasons:

- The run time for the CNN is too long (can be about 24 hours) and make tests difficult.
- The mfcc feature is compound of a 2d matrix in the shape of 300*13 - every second is 100 on 13 Mfcc coefficients. adopting this data into existing cnn code is complex due to the primarity of the number 13 which makes it hard to adopt it to the mnist cnn code.
- We made another attempt to implement a lstm RNN such that the previous steps memory is the mfcc feature and the next step is the label of this mfcc feature. This attempt failed because of inability to make the feature into three dimensions.

5 Conclusion

Our final test accuracy is about 75%. Since the guessing is about 20%, we gain 55% by using the softmax and adding multi-layer. In words, the computer is guessing right more than once on two. For the future work, and to improve the guessing, it should be interesting to add neural network, and adding also more data...
A little reservation: we think that the high accuracy of the test might be based on the fact that the train and the test are consist of the same records divided to short 10 second records. the data is shuffled and than divided into 70 percents of train and 30 percents of test. so, in the future we thought to use different data (records) for train and test.

References

[1] Diana Le Albert Chu Peter Lai. "Accent Classification of Non Native English Speakers". In: ().

[2] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, Boston, MA, 2006. ISBN: 978-0387310732.

[3] Peter Chien Phumchanit Watanaprakornkul Chantat Eksombatchai. "Accent Classification". In: ().