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. Our data-set includes each one of the next languages : French, Hebrew, USSR and English (UK and USA), five different recording.

Each recording 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

[✡] 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

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 using also neural network, we attempt to classify the speakers.

2 Approach

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 [1] 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."

2.1 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.

Adding of course the audio.

We decide to take only the native language of the fifth regions (France, USSR, UK, USA and Israel) and the audio that we convert into a WAV file. 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, we downloaded all the audio

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

in a 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, our 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.

2.2 Work description

To recognize accent, using Tensorflow, we implement a multinomial logistic regression (softmax). First, we recuperate 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.

We made some tests trying to improve the results, such as the next table 2.2.

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.

Table 2: Experiences

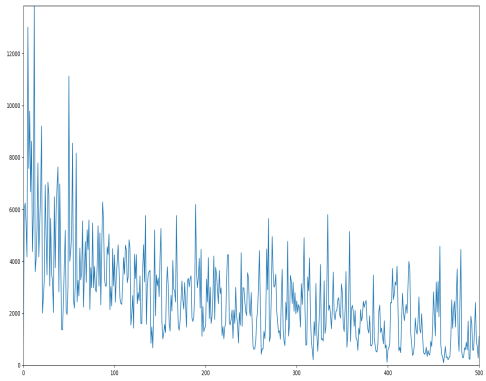
2.3 Multi-layer Perceptron

The classifier can have multiple hidden layers. In this project, we find that two or three hidden layers in general perform better. We choose the "adam" solver (optimizer) because it is an efficient stochastic gradient descent. For the activation function in the hidden layers, "relu" is chosen for its efficiency. The regularization penalty and initial learning rate are tuned.

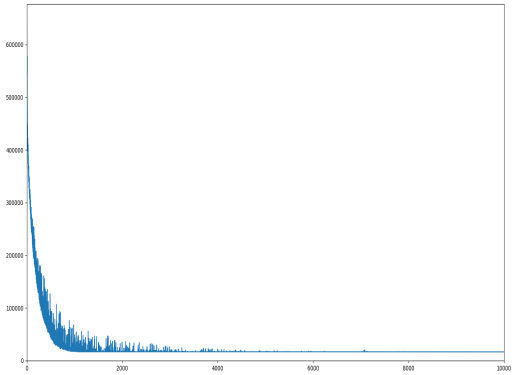
3 History and improvement

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

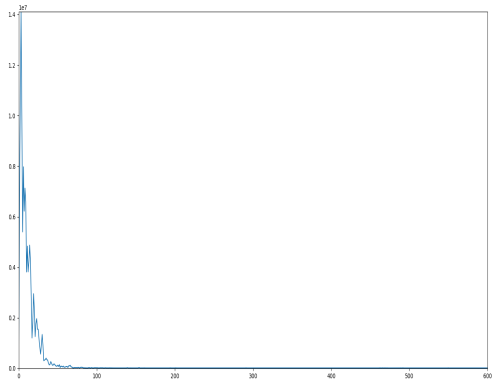
refer to all the experiences we made. Here are the picture of the loss function:



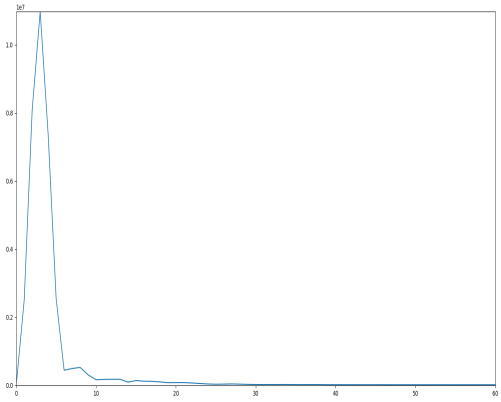
Notice than at this stage our accuracy was about 50%. Then, we add two hidden layers to our code, we improve our accuracy of 5%:



Notice that in particular on this picture, we try to change the epochs to 10000. To get our accuracy better, we decide to add regularization, such that we gain once again 5%, the loss function was like the next picture:



Finally, we decide to use the "relu" function for our hidden layer (we used add before), then, our accuracy improved to 75%, 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.

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 dataset. Here are the current parameter in use:
learning rate = 0.05
batch size = 600
number of steps = 60
hidden 1 = 256
hidden 2 = 256

4 Main result

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... Notice that in our current project, it's possible and even usual than a record use for the train and a record use for the test is made by the same recorder.

References

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