Aravind Natarajan

presentation: Chegg, 04/25.

Data Scientist at SunFunder.

Research Fellow (Physics) at Carnegie Mellon

Summer 2014: Data Science Fellow, Insight Data Science
created “Signs” to teach English through American Sign Language
for use in Deaf schools.

Fall 2014: Volunteer English teacher at Spituk monastery, Ladakh.

Spring 2015: Volunteer Data Scientist at DataKind, curating hospital
data relevant to the Ebola Virus Disease in Sierra-Leone.

Speaker Identification

The Problem:
Determine the speaker given **x** number of speakers and **y** seconds of speech.

The Data:
Open Speech and Language Resources:
LibriSpeech ASR Corpus: http://www.openslr.org/12/

- A corpus of nearly 1000 hours of speech at 16 khz.
- Each speaker ~ 30 minutes of speech.
- Has clean, as well as noisy speech samples.

example female speaker:

example female speaker:

example male speaker:

Converted .flac to .wav for 10 speakers: 5 female and 5 male.
Attempt to perform 10-class classification.
Uploaded the compressed audio folder to github.

example female speaker:

example male speaker:

Converted .flac to .wav for 10 speakers: 5 female and 5 male.
Attempt to perform 10-class classification.
Uploaded the compressed audio folder to github.

Q: What distinguishes one speaker from another?
A: *The frequency content.*

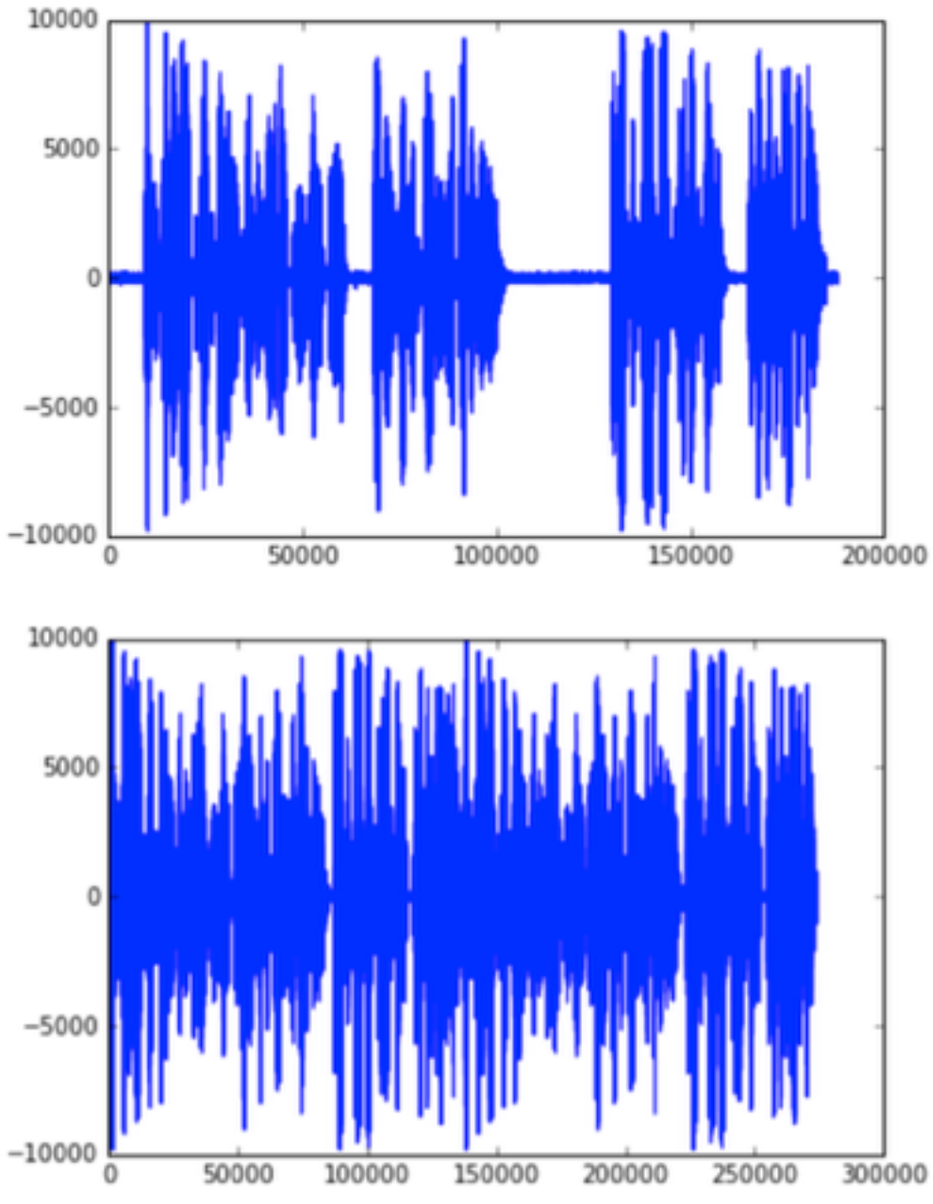
Nyquist's theorem:
Maximum frequency = *half* the sampling frequency.

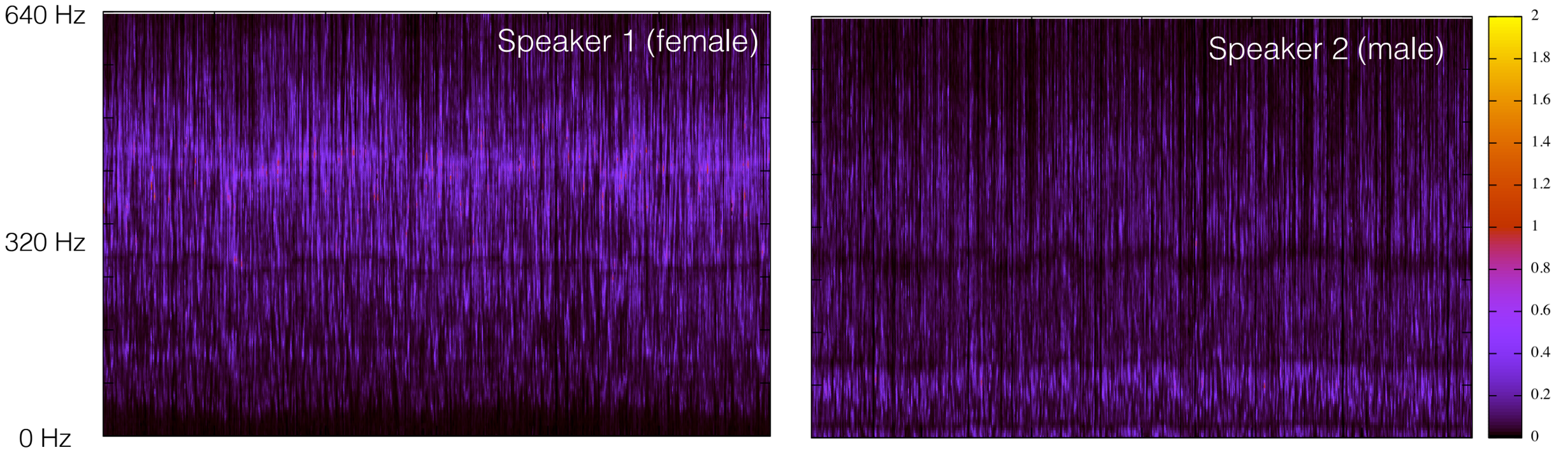
=> Max frequency = 8000 Hz!
But human voice has no frequencies that high.

We want to eliminate silent parts of speech:

Break the speech waveform into individual chunks of duration ΔT

Compute the root mean square value per chunk.
Ignore chunks with rms (chunk) < *n* * rms (speech)





For human speech, we need only 50 - 2000 Hz.
Lowest frequency = 1 / ΔT, Highest frequency = sampling frequency / 2
With ΔT = 0.2 s, frequency separation = 5 Hz.

Larger ΔT means we have more features! Easier to classify.
But large ΔT means it is harder to detect silent portions of speech.

audio files
converted to .wav files.
split into train, cv, and test samples.

Trained on about 20 minutes of audio.
25 - 1525 Hz -> 300 features.

Classification techniques:
(i) Gaussian modeling.
(ii) Machine Learning - e.g. a deep neural network.

(i) Gaussian modeling:

(a) Find the mean and standard deviation for each frequency component, for each speaker.

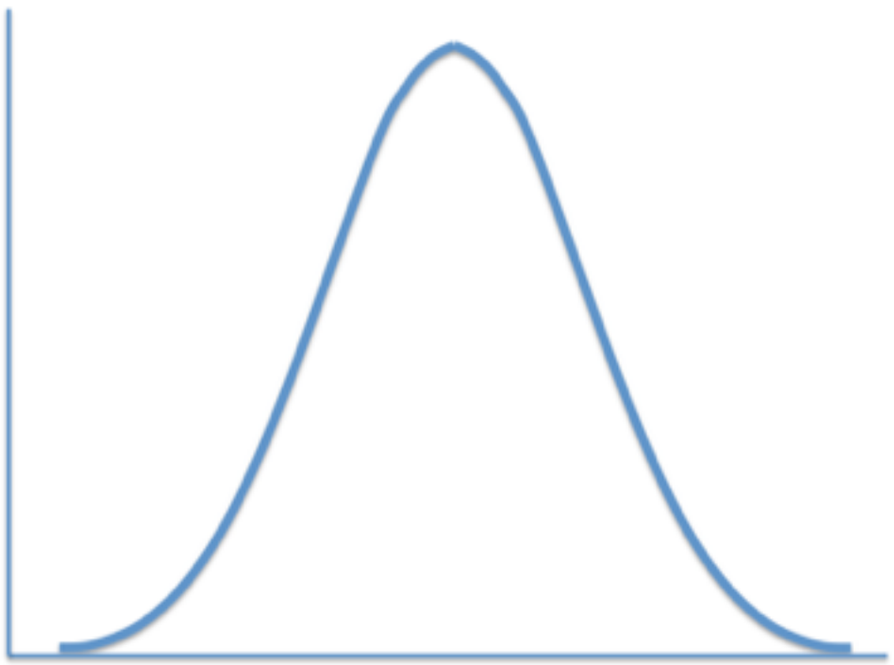
(b) Construct a gaussian (μ,σ) for each feature, for each speaker.

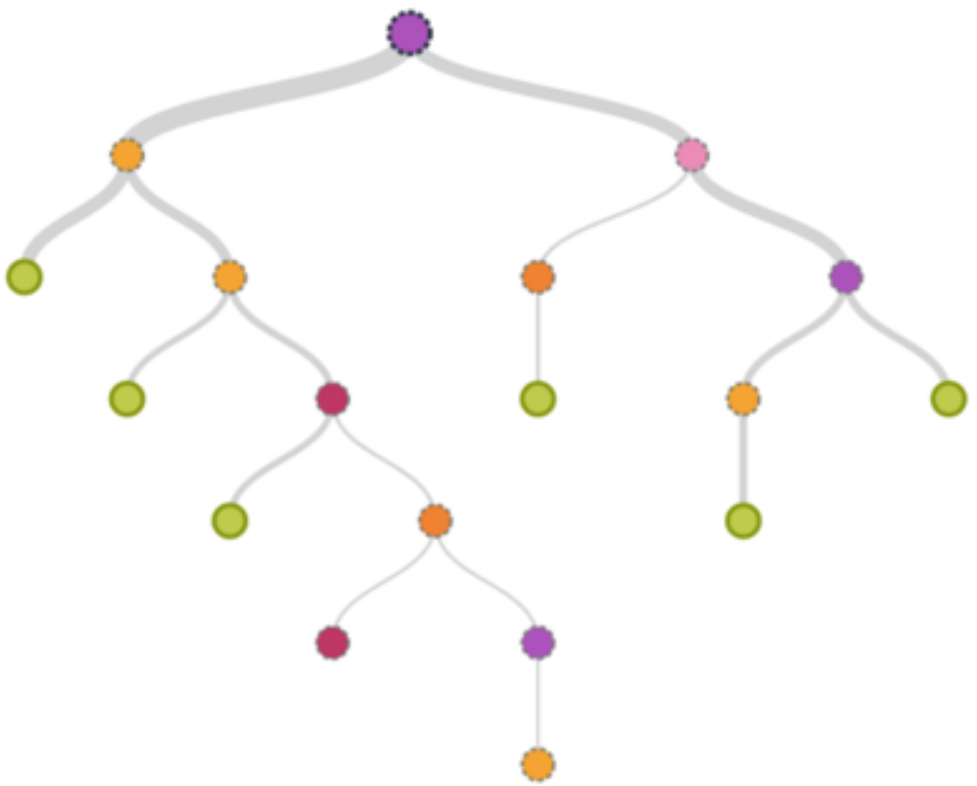
(c) Perform a Montecarlo simulation for each speaker x, for each frequency y:
Given the gaussian in (b), we draw random samples consistent with the distribution.
Each random sample is representative of the speaker x, at frequency y.

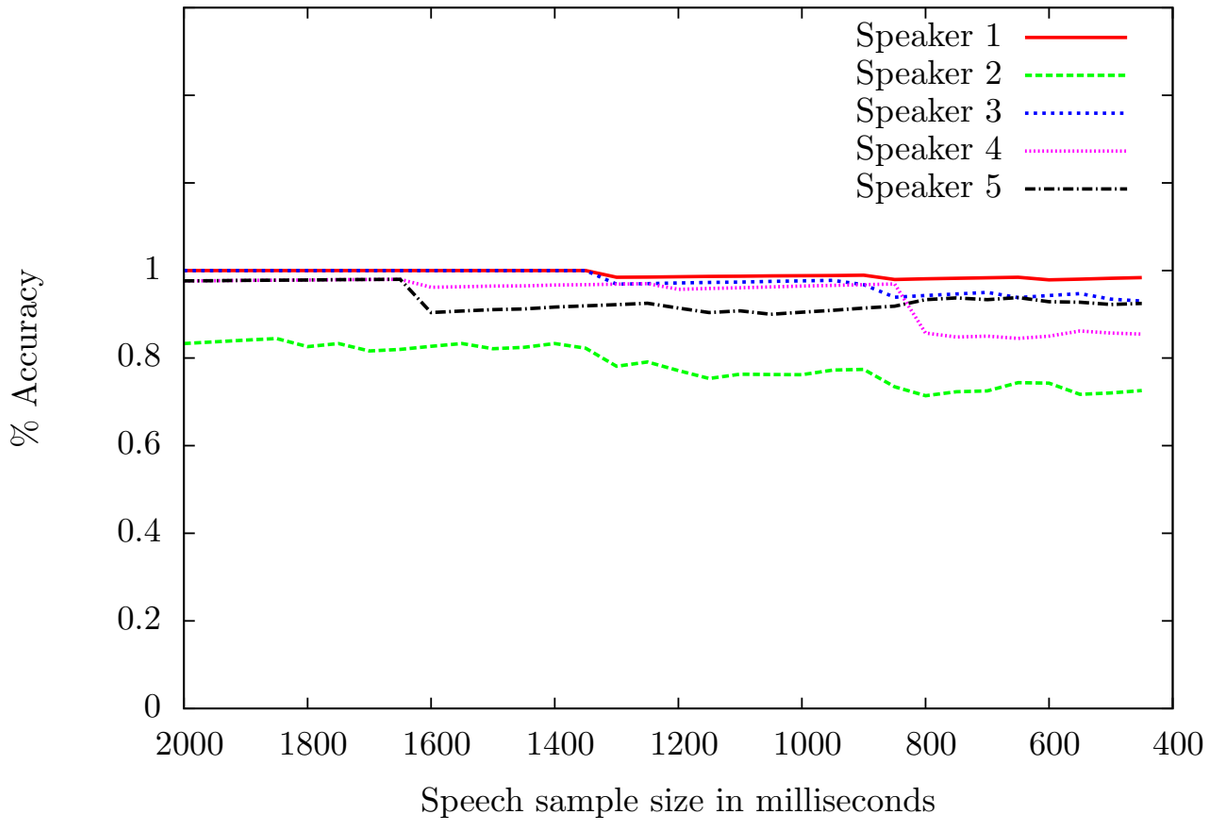
(d) Repeat (c) several thousand times - we now have a very large training set.

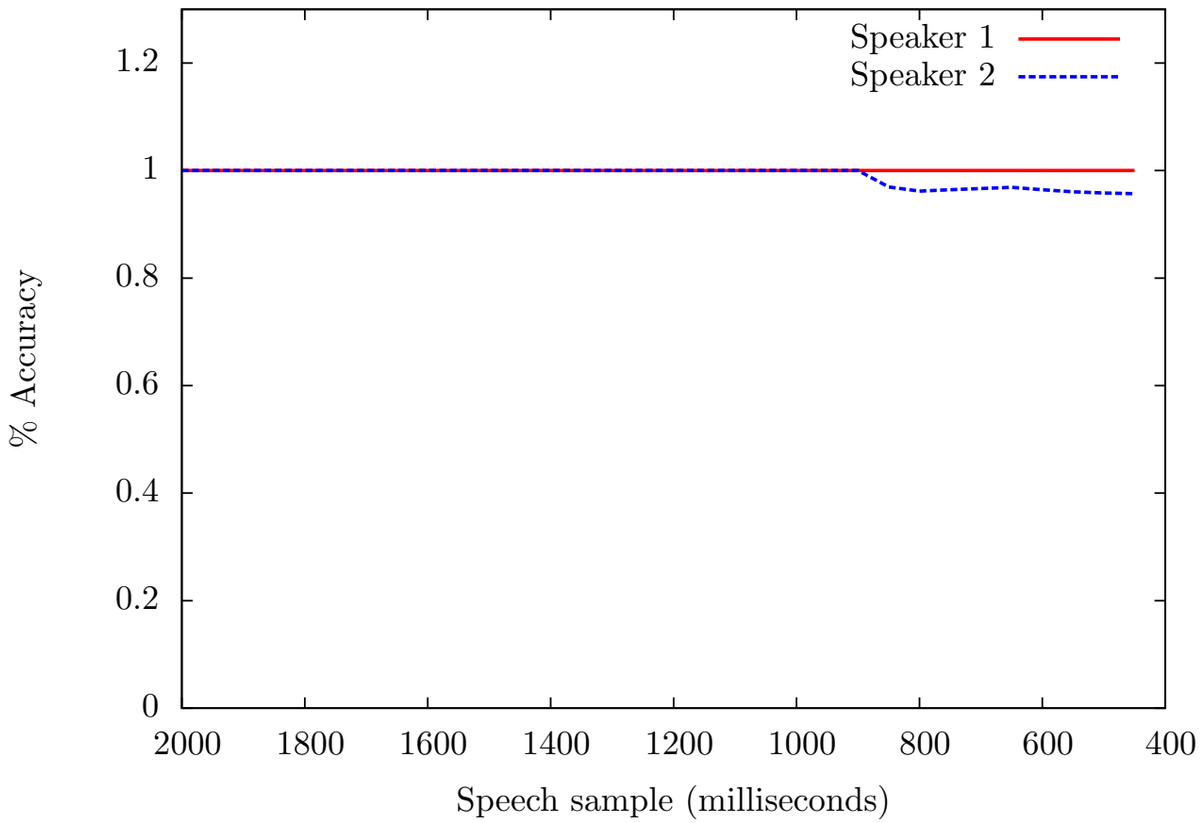
(e) Use a Decision Tree / Random Forest Classifier to identify useful features.

(f) Given a test set, find (μ,σ) and test the classifier.









It is a simple solution! And it works well for most speakers! (Speaker 2 doesn't do so great).

Trouble is - it doesn't scale well to *n* class classification where n > 5
We reduced a large array of numbers to just 2 values (μ,σ)

$\frac{(\mu_1 - \mu_2)}{\sqrt{\sigma_1 \sigma_2}}$ needs to be large.

Thus, we are throwing away a lot of data. Can we do better?

Rather than model the frequency components by a gaussian, we could feed all features to a classifier.

But what kind of classifier?

A neural network fits the problem well:

(i) It is easy for a human to distinguish speakers once the human has had time to study their voices.

(ii) It is difficult to deduce an algebraic boundary that distinguishes classes.

Which neural network package to use?

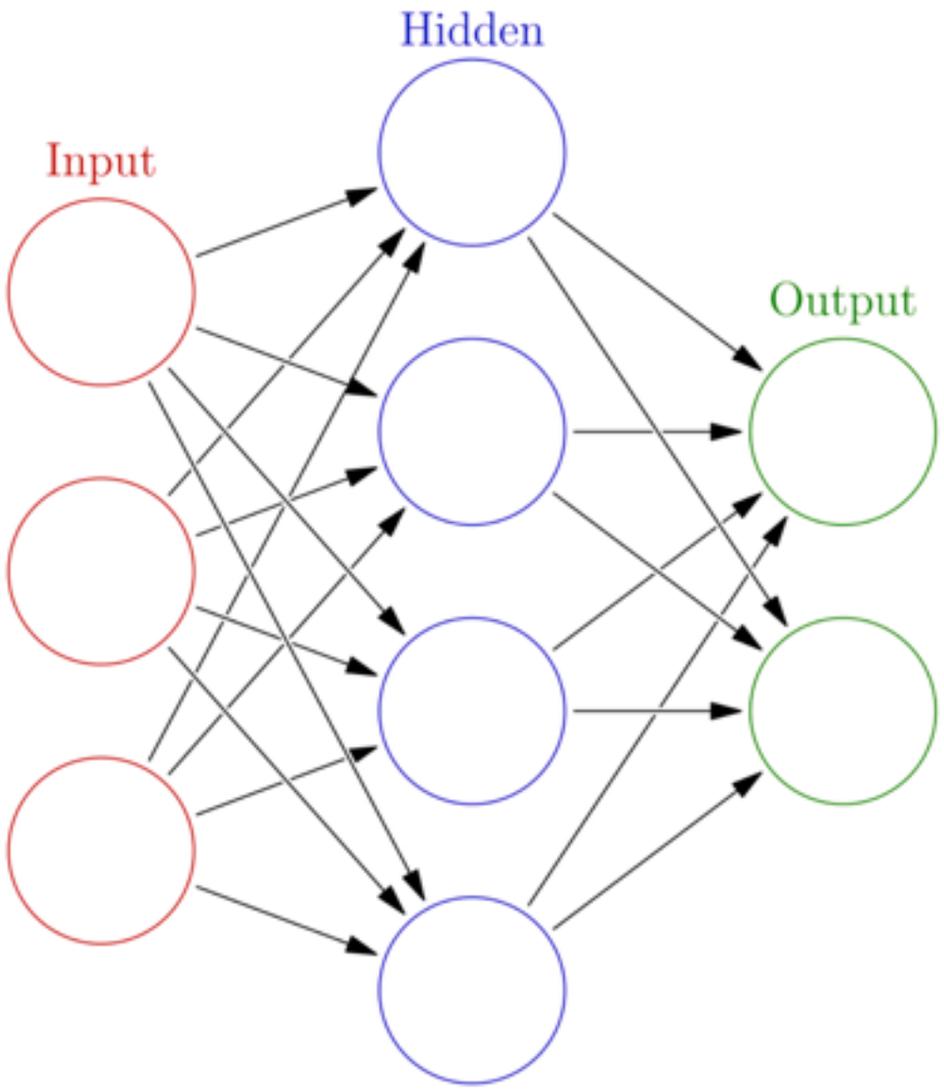
Scikit-Learn has a neural net implementation!

PyBrain

Theano

Tensorflow

For this problem, I have used PyBrain.

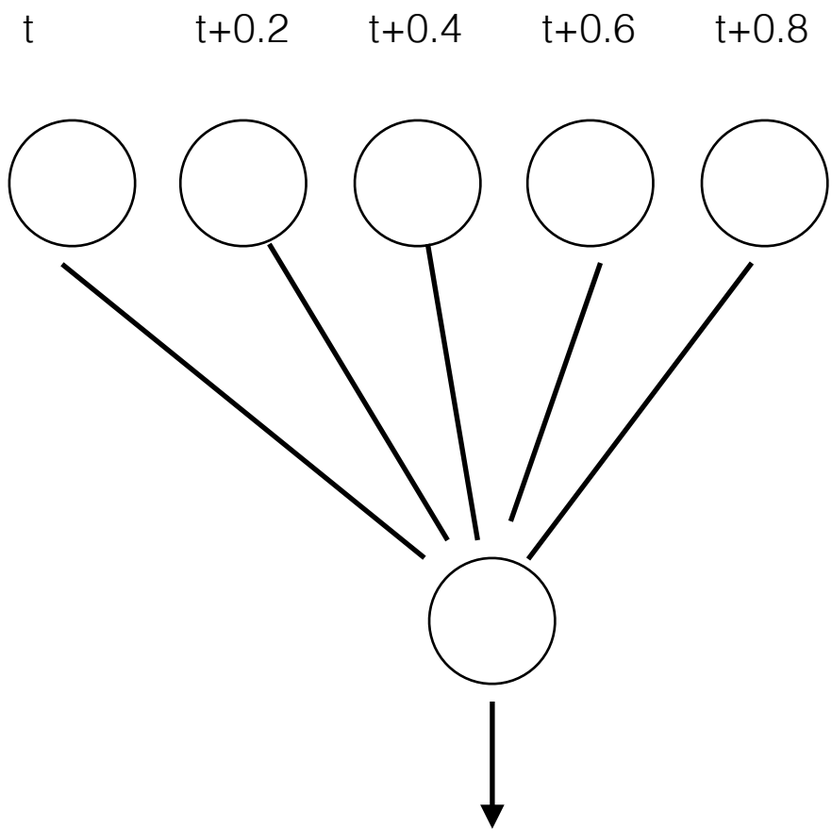


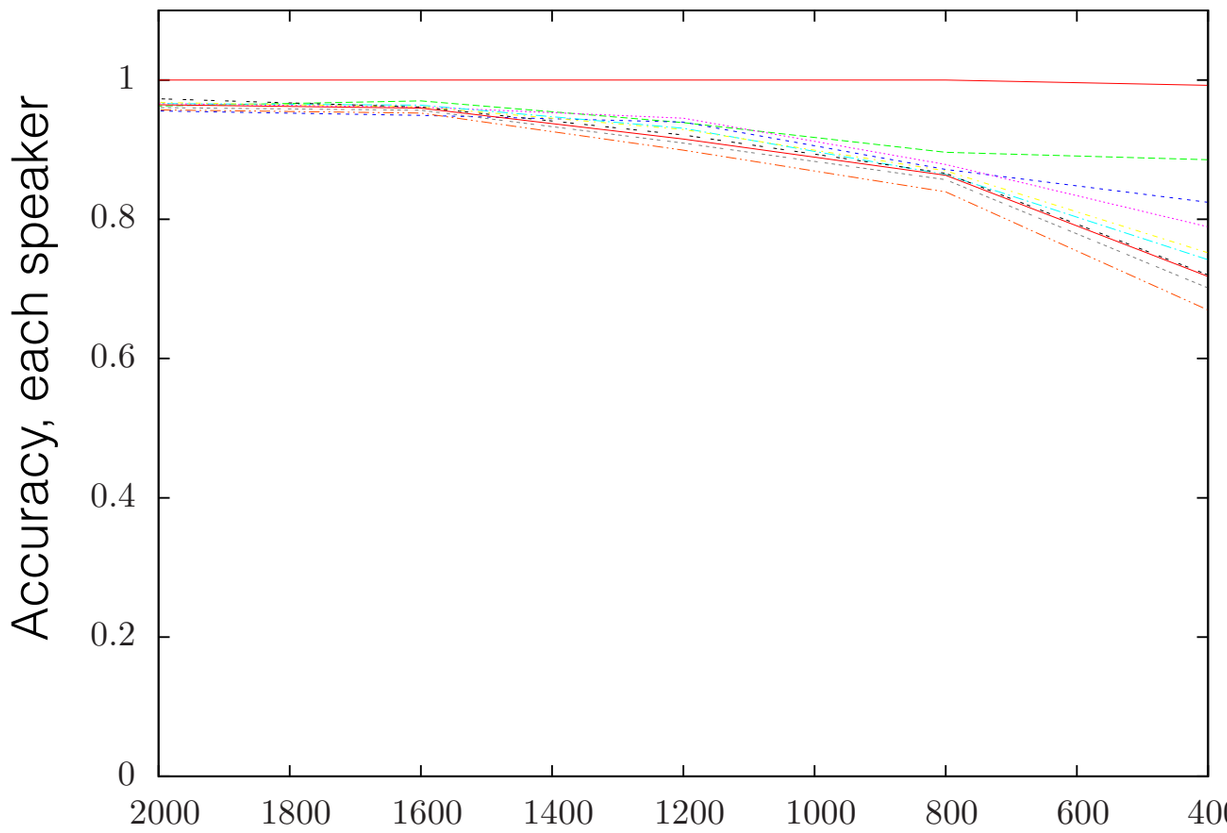
300 input neurons.
10 output neurons.

How many hidden neurons?

I like to start with the geometric mean of (input,output).
i.e. 50 neurons as a first estimate.

Let's try 10 neutrons per layer x 5 hidden layers.





Improvements:

1. More work on the NN architecture.

2. NN has a tendency to overfit. Better regularization techniques, e.g. dropout. Better feature selection pre-processing stages.