Speaker Recognition

Research, Applications, and Conclusions

Dmitriy Yakovlev

Department of Computer Science
Harvey Mudd College

December 17, 2008

Overview

Refresher on Speaker Recognition

Approaches

Discrete Fourier Transform

Linear Predictive Coding

Results

Overview

- Refresher on Speaker Recognition

Approaches

- Discrete Fourier Transform
- Linear Predictive Coding

Results

In General

Input: Audio encoding of speech
Output: Information that is used to classify speaker
Applications:

- User authentication
- Biometrics
- Caller ID
- . . .

The Problem I Approached

- **Speaker Identification**
  Identify person who spoke voice sample.
  - Population of seven people
  - Both male and female speakers
  - Text Independent
- **Tools**
  - MATLAB
  - Neural Network Toolbox
  - sound-recorder

Sample Collection

- **Direct Microphone input into MATLAB**
  - Impractical
  - Would have to store the samples somewhere, anyway...
  - Not implementable, we don't have the DSP toolbox installed.
- **Pre-recorded .wav files**
  - Portable
  - Collected via USB Microphone off-site
  - MATLAB has tools to work with wave files

Sample Collection

- **Direct Microphone input into MATLAB**
  - Impractical
  - Would have to store the samples somewhere, anyway...
  - Not implementable, we don't have the DSP toolbox installed.
- **Pre-recorded .wav files**
  - Portable
  - Collected via USB Microphone off-site
  - MATLAB has tools to work with wave files

Sample Details

- **7 subjects**
  - 4 females
  - 3 males
  - similar ages
- **Each subject records 1 minute of speech**
  - 16-bit
  - 1 channel
  - 44100 Hz sample rate
- Speech sample broken up into 5 10-second chunks
- Loaded into MATLAB via *wavread()*

What is it?

$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{2\pi i}{N} kn} \qquad k = 0, \ldots, N-1$$

- **Time Domain → Frequency Domain**
  - Implemented in MATLAB via *fft()*
  - Allows analysis of signals by their component frequencies
  - Feed frequencies into NN
- Sounds like it's ideal for voice work, right?
  - A person's voice has dominant frequencies
  - Text-independent by nature

What is it?

$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{2\pi i}{N} kn} \qquad k = 0, \ldots, N-1$$

- **Time Domain → Frequency Domain**
  - Implemented in MATLAB via *fft()*
  - Allows analysis of signals by their component frequencies
  - Feed frequencies into NN
- Sounds like it's ideal for voice work, right?
  - A person's voice has dominant frequencies
  - Text-independent by nature

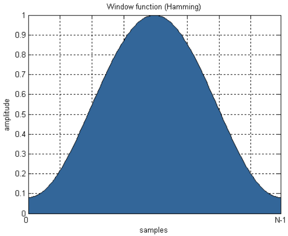
It's not.

- **Issues with DFT**
  - Too much noise - spectrum is very variable
  - Too much information - spectrum typically as big as sample size
  - Inconsistent - spectral amplitude dependent on volume
- **Solutions**
  - Noise filtering
  - Take only important, classifiable parts of spectrum
  - equalize volume of all samples prior to analysis
  - Window functions

It’s not.

- **Issues with DFT**
  - Too much noise - spectrum is very variable
  - Too much information - spectrum typically as big as sample size
  - Inconsistent - spectral amplitude dependent on volume
- **Solutions**
  - Noise filtering
  - Take only important, classifiable parts of spectrum
  - equalize volume of all samples prior to analysis
  - Window functions

Window functions



- Further divides sample signal into parts
- Amplifies inconsistency between different parts of one signal

In the end, DFT's are impractical for the problem.

- Fiddly
- Difficult to extract the *right* information
- Accuracy anywhere from 25% to 60%

Overview

Refresher on Speaker Recognition

Approaches

Discrete Fourier Transform

Linear Predictive Coding

Results

What is LPC?

Linear Predictive Coding is a "tool for representing the spectral envelope of speech in compressed form".

- Originally used for compression and data transfer by early communications networks
- Currently in use by phone companies for cell network data transfer

What does LPC give us?

- Time signal $\rightarrow$ Polynomial coefficients that represent it
- An approximation can be reproduced from the polynomial coefficients
- MATLAB's function *lpc()* simplifies the implementation

Getting LPC to work

- To produce a good polynomial approximation, LPC must be applied to frames
- 30-40 frames per second recommended for good speech reproduction

In my implementation, I used window functions to create these frames.

My approach

- Generate time series from .wav files
- Move through each signal, 1500 samples at a time
- Generate LPC coefficients for 1500 sample window
- average all LPC coefficients for a sample together

This creates a fairly unique representation for a signal.

Results

- Not very good...
- LPC on averaged windows has terrible accuracy. LPC on the whole signal (10 seconds with a 32-degree polynomial!) had better accuracy, even though that makes little sense.
- This suggests that averaging windows does not yield a good characteristic of a voice.

More Results

Trained data:

- Accuracy on windowed LPC: **40**% of speaker files properly identified
- Accuracy on non-windowes LPC: **71**% of speaker files properly identified

Untrained data:

- Accuracy on windowed LPC: **32**% of speaker files properly identified
- Accuracy on non-windowes LPC: **45**% of speaker files properly identified

Future Goals

- Improve the LPC windowing to work
- Investigate whether averaging LPC results is a good idea
- Investigate Neural Network issues
- Implement SOM-based voice recognition, not just MLP.