AMATH 582 Final Project

Pavel Zelinsky

Github: <https://github.com/palpalych/VoicesData>

**Abstract**

I am taking an in depth look into the VOiCES data set and try to use PCA to extract voice data from a noisy data set, as well as classify human speech from non-speech

## Introduction and Overview

The VOiCES data set is composed of hundreds of short recordings. In these recordings is an assortment of people reading passages, people reading passages with different background sounds, and the background sounds on their own. With this data I have two different goals. First, I want to take the speech only data and by using PCA to extract the speech data out of a noisy data set. Second, I want to write a model that can classify whether a recording is of human speech or of noise. I expect the classification to be particularly challenging for the speech with noise portion of the data set

## Theoretical Background

The SVD is a matrix decomposition that breaks any matrix A into two unitary parts (U and V) and a diagonal part ∑

The most important part of the SVD for us here is that if your data is put in as each column of A is a data point, then the first K columns of U\* can be used as a transform for your data into K principal components, with the rows of V cut off at the (K+1)th column are the projections of your data into the principal components:

Given that U and V are unitary, these components are orthogonal to each other. This helps split any data set into the most relevant parts. Note: if your data is instead inserted as rows of A, then everything is still true except U and V are reversed for which is the transform and which is the projections of A. Any new data can then also be projected into the same space via:

Where p is the projection of x into the V space. Here ∑-1 is trivial to compute since it is a diagonal matrix. Next we can reverse the process in order to see what the projection p looks like in the original A space:

I’m no longer calling it x because it won’t be the same – we reduced the dimensionality of the system when we went to the V space so we have lost some data, but it is an approximation of x. Putting the formulas for p and y together we get

This would seem like it should be y=x since U is unitary, but although U is unitary UK is not. This should be immediately clear since UK\*UK and UKUK\* won’t even have the same dimensionality. The idea for using this for noise reduction is that we create the SVD from A composed of only human speech, then when we apply this UKUK\* approximation to new noisy data we should extract only the principal components associated with human speech

## Algorithm Implementation and Development

The main backbone of the algorithm to remove noise is the SVD, for this I am using the built in Matlab functionality. But even before doing the SVD the data set can be massaged so that the SVD is more effective. Normally for sound data I would use a spectrograph, but the main issue is that I need to be able to invert any changes I make to the signal before computing the SVD since the only way to validate whether I have removed the noise from the sound is to listen to it. The spectrograph seems to have some partial inverse algorithms, but it is safer to instead use the FFT since that has a well-known and understood inverse. For validating the success of the algorithm I put aside 3 types of noisy data: person whose voice wasn’t part of the training set, person whose voice was part of the training set, and person reading the same text they read in the training set. For all of these I had data with the noise being background music, babble, and a TV

For the classification portion of the algorithm I am free to use the spectrogram and of course I used that since it provides a variety of information about sound. For the classification itself I used the Naïve Bayes algorithm, and for the data I ensured that I had roughly equal counts of each type (human speech, music, TV, babble) and randomly split it into 80% training set and 20% test set. I also set aside some data to never be part of the test or training set for additional validation

## Computational Results

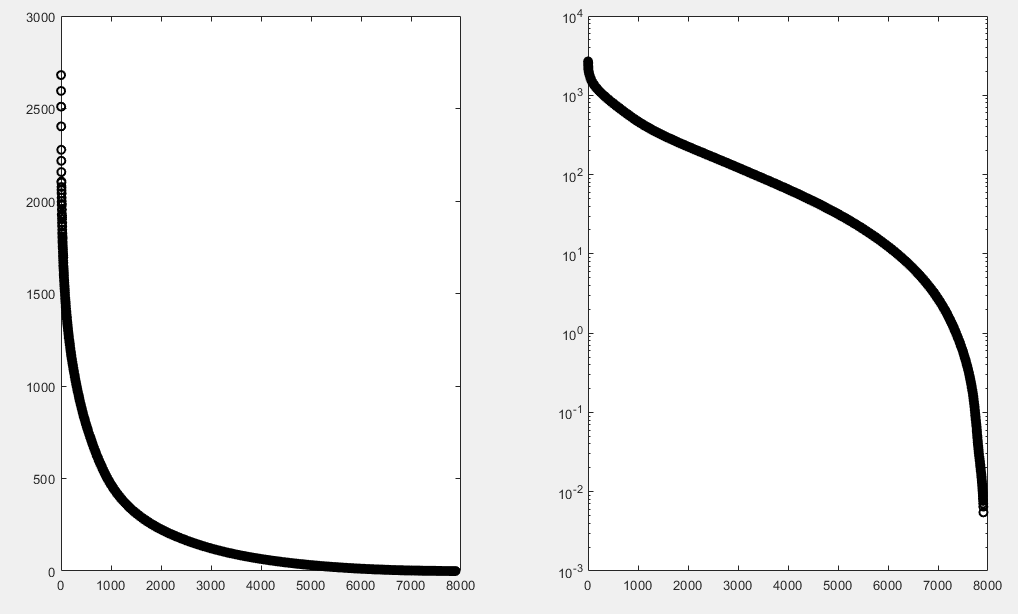
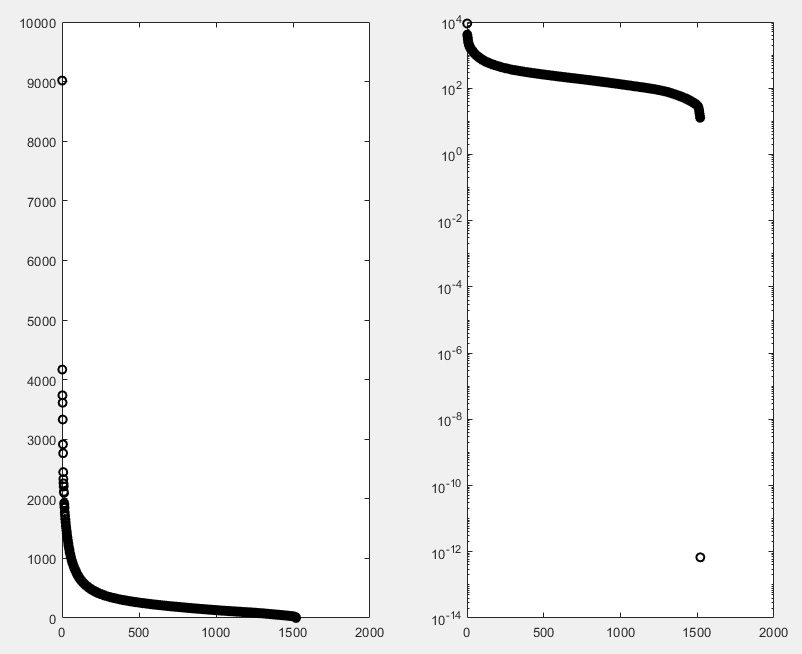
First taking a look at the results of noise reduction from speech. When putting together the speech only data the variances (Figure 1) don’t seem to have any strong indicators of good break points for principal components. The next thing to do is to listen to the approximations for different modes. When listening to the first order approximation I would hear a snippet of human voice from the training set, but no matter which of the columns of the approximations I would listen to it was the same snippet with some volume differences. Taking low rank approximations around 5 or 10 provided that same snippet of human voice from the training set but more in the background and with more general noise. As I would start taking 500 modes or more I would start hearing the voices of the actual snippet being approximated but with a fair amount of background noise. Note that these are from the training set where there is no noise, so the approximation is what is generating the noise. Beyond that increasing the number of modes continues to reduce the background noise and highlight the person’s voice, and as I got past 6000 modes out of 7908 the noise diminished enough to be unnoticeable. This is all well and good, but it’s approximations of already clean data

Figure 1: Clean Voice Data Variance on a normal and logarithmic scale

The next step is to of course run against noisy data to see whether this approach works. When using all the modes from the training set, the approximation had virtually no effect on any of the noisy data sets, even ones where the speaker hadn’t been heard by the system before. As I would reduce the number of modes, instead of making any particular portion of the sound snippets more prevalent it added white noise to the system. Further reducing the modes used for approximation only increased the white noise, up until around the 1000 mode mark the white noise became more prevalent than the other sounds. One interesting note is that during sections where in the original noisy signal either the noise temporarily stopped or the voice temporarily stopped, the approximated signal white noise cleared up as well. This is implying that what is happening here is that the noise is approximating to enough of the other human speech principal components that it’s intersecting badly with the actual human speech and causing the white noise. Unfortunately this means that for this purpose, using these approximations is quite useless in trying to remove the noise from the data

Figure 2: Voice and Noise data Variance on a normal and logarithmic scale

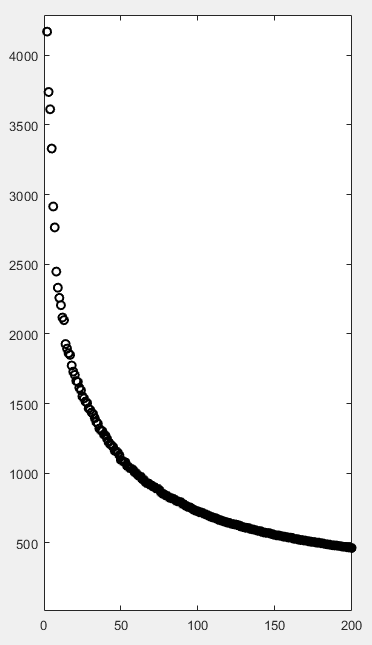


Figure 3: Voice and Noise data Variance

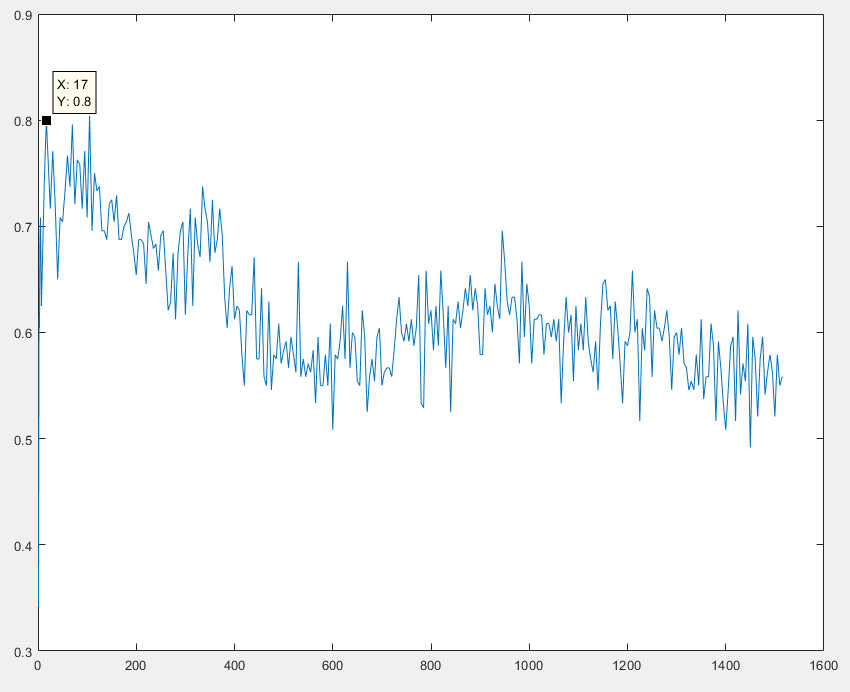
Next, taking a look at categorizing speech vs. different types of noise. We first look at the variances (Figure 2), and it seems that there is a fast drop-off within the first 200 modes. With closer inspection (Figure 3) there are gaps in modes (around 7, 13, and 17) but nothing particularly stands out. Because of this I ran the training algorithm on these particular modes of interest and every 5th mode beyond that in order to test what kind of prediction accuracy I get for the test set (Figure 4). Note that in this I would randomize the test and training set for every mode. One interesting note is that mode 17 was in fact one of the highest accuracy results, but this was from a single run so although there is some merit to this data it could be luck. One pattern that is consistent though is that taking more than 350 modes is overfitting the data and reduces the accuracy. Given that, Figure 5 is averaging 5 runs for any particular number of modes under 400. Although around 100 modes is the best in Figure 5, I chose 85 modes as one of the best options since this data is not perfectly accurate and the general trend of the graph is that there is a local maximum somewhere between 70 and 100

Figure 4: Effects of number of modes used on the model accuracy

Now that I have an optimal maximum mode to use, I can do the same process for the minimum mode to use as well. Figure 6 very clearly shows that starting with anything other than the first mode is unwise

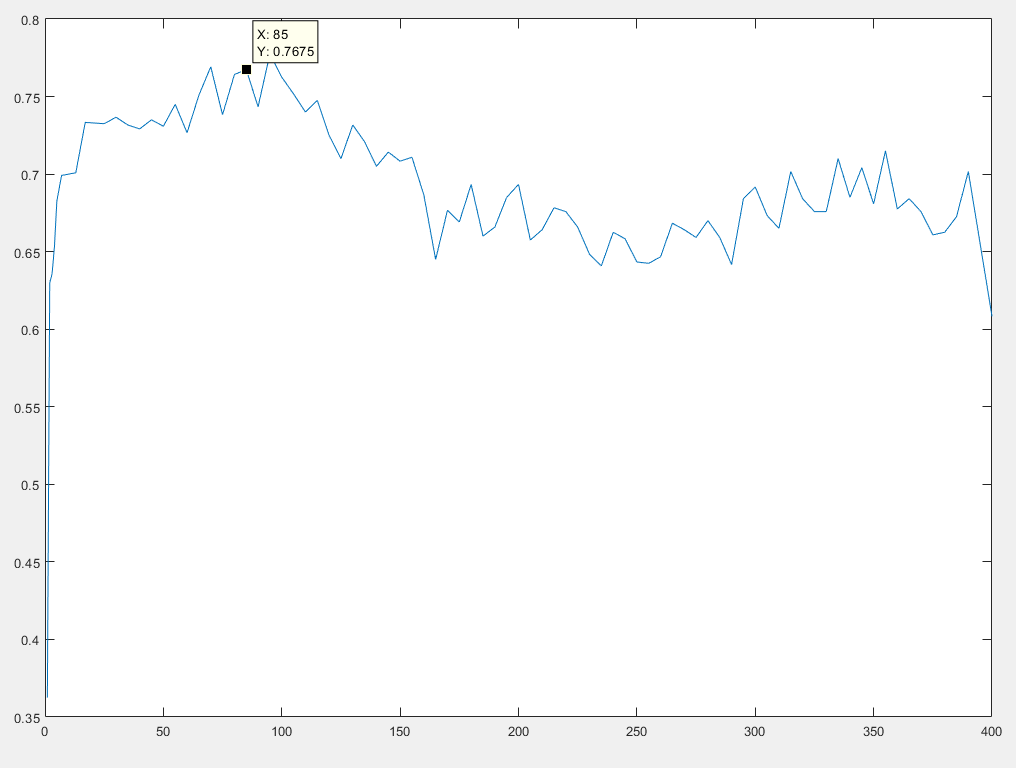
With the best set of modes to use, I can finally run my validation set against the model. Surprisingly, despite consistently getting around 80% accuracy for the test set, for the validation set I was always getting 61.25% accuracy. This number seems somewhat low, but a big note is that for every sound snippet it is making a prediction for whether something is voice, music, TV, or babble, so if it was random guessing the expected accuracy is 25%. That said, where is the algorithm failing? Figure 7 highlights all the incorrectly made predictions. The biggest surprise is that the algorithm never thinks anything is the TV and thus gets the TV classification wrong every time – this implies that potentially the TV shares too many commonalities with babble and music for the model to distinguish between them. A major success of this model is that the highlighted in red predictions are the only ones where it was wrong for whether something was voice vs. noise, where noise is any of TV, music, or babble. This means that if the goal was to only distinguish between speech and non-speech, this model has 97.5% accuracy! Note that this accuracy number is compared to the 50% chance of correctly guessing between the two options

Figure 5: Effects of number of modes used on the model accuracy average across multiple random training sets

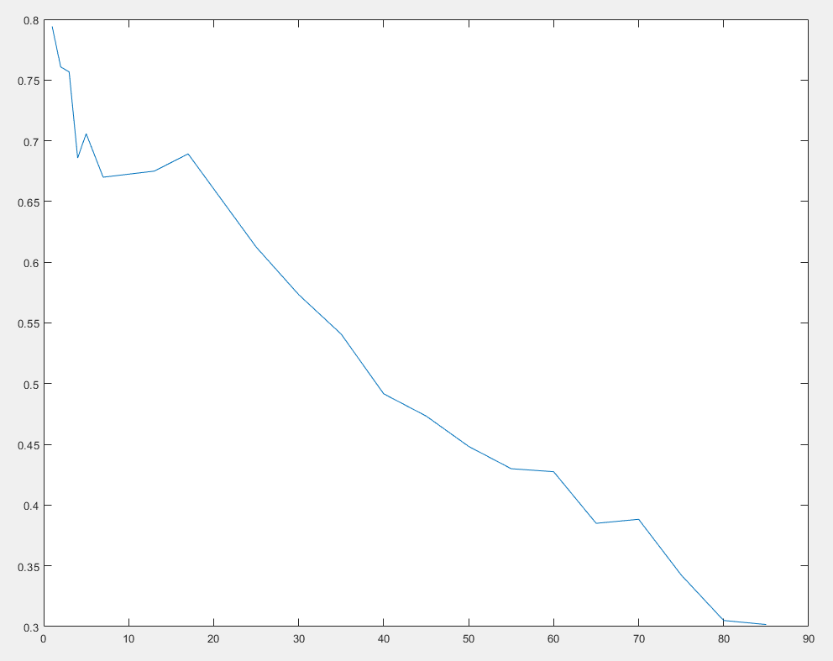


Figure 6: Effects of changing the starting mode on the model accuracy

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Truth** | **Babble** | **Music** | **TV** | **Voice** |
| Predictions for | 'babble' | 'music' | 'babble' | 'voice' |
| different samples | 'babble' | 'music' | 'babble' | 'voice' |
|  | 'babble' | 'music' | 'music' | 'voice' |
|  | 'babble' | 'babble' | 'music' | 'voice' |
|  | 'babble' | 'music' | 'music' | 'voice' |
|  | 'babble' | 'babble' | 'babble' | 'voice' |
|  | 'babble' | 'music' | 'music' | 'voice' |
|  | 'babble' | 'music' | 'babble' | 'voice' |
|  | 'babble' | 'babble' | 'music' | 'voice' |
|  | 'babble' | 'babble' | 'music' | 'voice' |
|  | 'babble' | 'music' | 'babble' | 'voice' |
|  | 'babble' | 'music' | 'music' | 'voice' |
|  | 'babble' | 'music' | 'music' | 'voice' |
|  | 'babble' | 'babble' | 'babble' | 'voice' |
|  | 'babble' | 'babble' | 'babble' | 'babble' |
|  | 'babble' | 'babble' | 'voice' | 'babble' |
|  | 'babble' | 'babble' | 'music' | 'voice' |
|  | 'babble' | 'music' | 'babble' | 'voice' |
|  | 'babble' | 'music' | 'music' | 'voice' |
|  | 'babble' | 'babble' | 'babble' | 'voice' |

Figure 7: Model predictions for different sound samples. Errors highlighted

## Summary and Conclusion

In conclusion, although the SVD can provide a projection space and an approximation from that projection space it does not provide an easy way to extract the main signal from a noisy snippet of data. The SVD can be successfully used as a way to reduce data sets to manageable sizes while keeping the relevant information for machine learning models.

References:

Voice Data set: <https://registry.opendata.aws/lab41-sri-voices/>

Voices Obscured in Complex Environmental Settings (VOICES) corpus by Colleen Richey and Maria A. Barrios and Zeb Armstrong and Chris Bartels and Horacio Franco and Martin Graciarena and Aaron Lawson and Mahesh Kumar Nandwana and Allen Stauffer and Julien van Hout and Paul Gamble and Jeff Hetherly and Cory Stephenson and Karl Ni, published in 2018

<https://arxiv.org/abs/1804.05053>

Appendix A – MATLAB Functions and Code Notes:

Semilogy – plot on a logarithmic scale

Pcolor, shading, set, xlabel, ylabel, title, colormap, figure, subplot – functions used for plotting, organizing, and labeling the data

Svd – best function in Matlab. Computes the SVD. The ‘econ’ flag is especially useful

Diag – extract just the diagonal elements of a matrix, useful when handling the SVD

Randperm – generate a random permutation vector. This is helpful for randomly rearranging data for a training and test set

Fitcnd, predict – train a Naïve Bayes model and make a prediction from the model

Appendix B – MATLAB Code:

FinalProject.m

%% Start with a clean slate

clear all; close all; clc

%% Trying to remove noise from recording project

%% load all data

directories = dir('E:\FakeDesktop\voices\VOiCES\_devkit\distant-16k\speech\train\rm1\none');

allVoicesNoNoiseFFT = zeros(8000,7908);

index = 1;

for i=1:length(directories)

if directories(i).name == "." | directories(i).name == ".."

continue

end

directories(i).name

voices = dir(strcat(directories(i).folder, '\', directories(i).name));

for j=1:length(voices)

if voices(j).name == "." | voices(j).name == ".."

continue

end

[voice,Fs] = audioread(strcat(voices(j).folder, '\', voices(j).name));

for k=1:floor(length(voice)/Fs)

if k > 10

break

end

allVoicesNoNoiseFFT(:,index) = fft(voice(1+Fs\*(k-1):Fs\*k-Fs/2));

index = index + 1;

end

end

end

%% compute SVDs

mean\_allVoicesFFT = mean(allVoicesNoNoiseFFT);

allVoicesNoNoiseFFT = allVoicesNoNoiseFFT - mean\_allVoicesFFT;

[uf,sf,vf] = svd(allVoicesNoNoiseFFT,'econ');

%% variances

sigf=diag(sf);

figure(2)

subplot(1,2,1), plot(sigf,'ko','Linewidth',[1.5])

subplot(1,2,2), semilogy(sigf,'ko','Linewidth',[1.5])

%% what do some of the modes sound like for the data?

modes = 7000;

approx1 = ifft(uf(:,1:modes)\*sf(1:modes,1:modes)\*vf(:,1:modes)' + mean\_allVoicesFFT);

p = audioplayer(1\*approx1(:,10),Fs);

p.play();

%p = audioplayer(ifft(allVoicesNoNoiseFFT(:,1)),Fs);

%p.play();

%% load noisy data

%noisyPath = 'E:\FakeDesktop\voices\VOiCES\_devkit\distant-16k\speech\train\rm1\musi\sp0093\Lab41-SRI-VOiCES-rm1-musi-sp0093-ch123172-sg0024-mc01-stu-clo-dg020.wav';

noisyPath = 'E:\FakeDesktop\voices\VOiCES\_devkit\distant-16k\speech\train\rm1\musi\sp0083\Lab41-SRI-VOiCES-rm1-musi-sp0083-ch009960-sg0031-mc01-stu-clo-dg110.wav';

%noisyPath = 'E:\FakeDesktop\voices\VOiCES\_devkit\distant-16k\speech\train\rm1\musi\sp8713\Lab41-SRI-VOiCES-rm1-musi-sp8713-ch296159-sg0045-mc05-stu-far-dg110.wav';

[noisyVoice,noisyFs] = audioread(noisyPath);

noisyVoiceFFT = zeros(8000,floor(length(noisyVoice)/noisyFs));

for k=1:floor(length(noisyVoice)/noisyFs)

noisyVoiceFFT(:,2\*k-1) = fft(noisyVoice(1+noisyFs\*(k-1):noisyFs\*k-Fs/2));

noisyVoiceFFT(:,2\*k) = fft(noisyVoice(1+noisyFs\*k-Fs/2:noisyFs\*k));

end

mean\_noisyFFT = mean(noisyVoiceFFT);

noisyVoiceFFT = noisyVoiceFFT - mean\_noisyFFT;

%% can we extract noiseless data from FFT?

startmode = 1;

endmode = 4000;

Transform = uf(:,startmode:endmode)';

inverseTransform = uf(:,startmode:endmode);

transformedNoise = (Transform \* noisyVoiceFFT)';

revertTransformNoise = inverseTransform\*transformedNoise';

%p = audioplayer(noisyVoice, Fs);

%p.play();

%p = audioplayer(ifft(revertTransformNoise),Fs);

%p.play();

original = [];

for i=1:size(revertTransformNoise,2)

original = [original ifft(revertTransformNoise(:,i))];

end

noiseReducedTrack = reshape(original,size(original,1)\*size(original,2),1);

p = audioplayer(noiseReducedTrack,Fs);

p.play();

%%

cleanVersionPath = 'E:\FakeDesktop\voices\VOiCES\_devkit\distant-16k\speech\train\rm1\none\sp0083\Lab41-SRI-VOiCES-rm1-none-sp0083-ch009960-sg0031-mc01-stu-clo-dg110.wav';

%cleanVersionPath = 'E:\FakeDesktop\voices\VOiCES\_devkit\distant-16k\speech\train\rm1\sp8713\Lab41-SRI-VOiCES-rm1-none-sp8713-ch296159-sg0045-mc05-stu-far-dg110.wav';

[cleanVoice,cleanFs] = audioread(cleanVersionPath);

figure(2)

subplot(3,1,1)

plot(abs(cleanVoice));

set(gca,'Xlim',[0 250000])

title('No Noise Signal');

subplot(3,1,2)

plot(abs(noiseReducedTrack));

set(gca,'Xlim',[0 250000])

title('Attempted Noise Reduction');

subplot(3,1,3)

plot(abs(noisyVoice));

set(gca,'Xlim',[0 250000])

title('Noisy Signal');

p = audioplayer(noisyVoice,Fs);

p.play();

%%

figure(3)

subplot(3,1,1)

plot(abs(fftshift(fft(cleanVoice(1:Fs)))));

title('No Noise Signal');

subplot(3,1,2)

plot(abs(fftshift(fft(noiseReducedTrack(1:Fs)))));

title('Attempted Noise Reduction');

subplot(3,1,3)

plot(abs(fftshift(fft(noisyVoice(1:Fs)))));

title('Noisy Signal');

%% Categorize Human speech vs. noise

babbleFolder = dir('E:\FakeDesktop\voices\VOiCES\_devkit\distant-16k\distractors\rm3\babb');

%babbleSounds = [];

babbleSounds = zeros(57351,380);

index = 1;

% skip first 3 elements: '.', '..', and the cross validation data

for i=4:length(babbleFolder)

if babbleFolder(i).name == "." | babbleFolder(i).name == ".."

continue

end

% each file is 30+ minutes long. Take some 5 second chunks from each

[m,Fs] = audioread(strcat(babbleFolder(i).folder, '\', babbleFolder(i).name));

for j=1:20

babbleSounds(:,index) = reshape(spectrogram(m(j\*Fs\*30:2:j\*Fs\*30+Fs\*5+1,1)),8193\*7,1);

index = index + 1;

end

end

musicFolder = dir('E:\FakeDesktop\voices\VOiCES\_devkit\distant-16k\distractors\rm3\musi');

musicSounds = zeros(57351,380);

index = 1;

% skip first 3 elements: '.', '..', and the cross validation data

for i=4:length(musicFolder)

if musicFolder(i).name == "." | musicFolder(i).name == ".."

continue

end

% each file is 30+ minutes long. Take some 5 second chunks from each

[m,Fs] = audioread(strcat(musicFolder(i).folder, '\', musicFolder(i).name));

for j=1:20

musicSounds(:,index) = reshape(spectrogram(m(j\*Fs\*30:2:j\*Fs\*30+Fs\*5+1,1)),8193\*7,1);

index = index + 1;

end

end

tvFolder = dir('E:\FakeDesktop\voices\VOiCES\_devkit\distant-16k\distractors\rm3\tele');

tvSounds = zeros(57351,380);

index = 1;

% skip first 3 elements: '.', '..', and the cross validation data

for i=4:length(tvFolder)

if tvFolder(i).name == "." | tvFolder(i).name == ".."

continue

end

% each file is 30+ minutes long. Take some 5 second chunks from each

[m,Fs] = audioread(strcat(tvFolder(i).folder, '\', tvFolder(i).name));

for j=1:20

tvSounds(:,index) = reshape(spectrogram(m(j\*Fs\*30:2:j\*Fs\*30+Fs\*5+1,1)),8193\*7,1);

index = index + 1;

end

end

directories = dir('E:\FakeDesktop\voices\VOiCES\_devkit\distant-16k\speech\train\rm1\none');

allVoicesNoNoiseSPCTR = zeros(57351,380);

index = 1;

for i=1:length(directories)

if directories(i).name == "." | directories(i).name == ".."

continue

end

%directories(i).name

voices = dir(strcat(directories(i).folder, '\', directories(i).name));

for j=1:2:length(voices)

if voices(j).name == "." | voices(j).name == ".."

continue

end

[m,Fs] = audioread(strcat(voices(j).folder, '\', voices(j).name));

allVoicesNoNoiseSPCTR(:,index) = reshape(spectrogram(m(Fs\*1:2:Fs\*6+1,1)),8193\*7,1);

index = index + 1;

if index == 381

break;

end

end

if index == 381

break;

end

end

allSounds = abs([babbleSounds musicSounds tvSounds allVoicesNoNoiseSPCTR]);

mean\_allSounds = mean(allSounds);

allSoundsCentered = allSounds - mean\_allSounds;

[u,s,v] = svd(allSoundsCentered,'econ');

sig=diag(s);

figure(6)

subplot(1,2,1), plot(sig,'ko','Linewidth',[1.5])

subplot(1,2,2), semilogy(sig,'ko','Linewidth',[1.5])

%%

endmodelist = [1 2 3 4 5 7 13 17 25:5:1515];

xexpected=string(zeros(240,1));

xexpected(1:60)="babble";

xexpected(61:120)="music";

xexpected(121:180)="tv";

xexpected(181:end)="voice";

ctrain=string(zeros(320\*4,1));

ctrain(1:320)="babble";

ctrain(321:640)="music";

ctrain(641:960)="tv";

ctrain(961:end)="voice";

trainAccuracies = zeros(1,length(endmodelist));

for i=1:length(endmodelist)

startmode = 1;

endmodes = endmodelist(i);

random\_babble = randperm(380);

random\_music = randperm(380);

random\_tv = randperm(380);

random\_voice = randperm(380);

babble = v(1:380,startmode:endmodes);

music = v(381:760,startmode:endmodes);

tv = v(761:1140,startmode:endmodes);

voice = v(1141:1520,startmode:endmodes);

xtrain=[babble(random\_babble(1:320),:); music(random\_music(1:320),:); tv(random\_tv(1:320),:); voice(random\_voice(1:320),:)];

xtest=[babble(random\_babble(321:end),:); music(random\_music(321:end),:); tv(random\_tv(321:end),:); voice(random\_voice(321:end),:)];

nb=fitcnb(xtrain,ctrain);

trainAccuracies(i) = sum(nb.predict(xtest)==xexpected)/length(xexpected);

end

figure(7)

plot(endmodelist, trainAccuracies);

%%

endmodelist = [1 2 3 4 5 7 13 17 25:5:400];

xexpected=string(zeros(240,1));

xexpected(1:60)="babble";

xexpected(61:120)="music";

xexpected(121:180)="tv";

xexpected(181:end)="voice";

ctrain=string(zeros(320\*4,1));

ctrain(1:320)="babble";

ctrain(321:640)="music";

ctrain(641:960)="tv";

ctrain(961:end)="voice";

trainAccuracies = zeros(1,length(endmodelist));

for j=1:5

for i=1:length(endmodelist)

startmode = 1;

endmodes = endmodelist(i);

random\_babble = randperm(380);

random\_music = randperm(380);

random\_tv = randperm(380);

random\_voice = randperm(380);

babble = v(1:380,startmode:endmodes);

music = v(381:760,startmode:endmodes);

tv = v(761:1140,startmode:endmodes);

voice = v(1141:1520,startmode:endmodes);

xtrain=[babble(random\_babble(1:320),:); music(random\_music(1:320),:); tv(random\_tv(1:320),:); voice(random\_voice(1:320),:)];

xtest=[babble(random\_babble(321:end),:); music(random\_music(321:end),:); tv(random\_tv(321:end),:); voice(random\_voice(321:end),:)];

nb=fitcnb(xtrain,ctrain);

trainAccuracies(i) = trainAccuracies(i) + sum(nb.predict(xtest)==xexpected)/length(xexpected);

end

end

figure(8)

plot(endmodelist, trainAccuracies/5);

%% startmode

startmodelist = [1 2 3 4 5 7 13 17 25:5:85];

xexpected=string(zeros(240,1));

xexpected(1:60)="babble";

xexpected(61:120)="music";

xexpected(121:180)="tv";

xexpected(181:end)="voice";

ctrain=string(zeros(320\*4,1));

ctrain(1:320)="babble";

ctrain(321:640)="music";

ctrain(641:960)="tv";

ctrain(961:end)="voice";

trainAccuracies = zeros(1,length(startmodelist));

for j=1:5

for i=1:length(startmodelist)

startmode = startmodelist(i);

endmodes = 85;

random\_babble = randperm(380);

random\_music = randperm(380);

random\_tv = randperm(380);

random\_voice = randperm(380);

babble = v(1:380,startmode:endmodes);

music = v(381:760,startmode:endmodes);

tv = v(761:1140,startmode:endmodes);

voice = v(1141:1520,startmode:endmodes);

xtrain=[babble(random\_babble(1:320),:); music(random\_music(1:320),:); tv(random\_tv(1:320),:); voice(random\_voice(1:320),:)];

xtest=[babble(random\_babble(321:end),:); music(random\_music(321:end),:); tv(random\_tv(321:end),:); voice(random\_voice(321:end),:)];

nb=fitcnb(xtrain,ctrain);

trainAccuracies(i) = trainAccuracies(i) + sum(nb.predict(xtest)==xexpected)/length(xexpected);

end

end

figure(9)

plot(startmodelist, trainAccuracies/5);

%% retrain model with best start and end

startmode = 1;

endmodes = 85;

Transform = s(startmode:endmodes,startmode:endmodes)\u(:,startmode:endmodes)';

random\_babble = randperm(380);

random\_music = randperm(380);

random\_tv = randperm(380);

random\_voice = randperm(380);

babble = v(1:380,startmode:endmodes);

music = v(381:760,startmode:endmodes);

tv = v(761:1140,startmode:endmodes);

voice = v(1141:1520,startmode:endmodes);

xtrain=[babble(random\_babble(1:320),:); music(random\_music(1:320),:); tv(random\_tv(1:320),:); voice(random\_voice(1:320),:)];

nb=fitcnb(xtrain,ctrain);

%%

correctBayes = 0;

% take a completely unused Music noise sample and test against it

[m,Fs] = audioread('E:\FakeDesktop\voices\VOiCES\_devkit\distant-16k\distractors\rm3\musi\Lab41-SRI-VOiCES-rm3-musi-mc01-stu-clo.wav');

testMusic = zeros(8193\*7, 20);

for j=1:20

testMusic(:,j) = reshape(spectrogram(m(j\*Fs\*30:2:j\*Fs\*30+Fs\*5+1,1)),8193\*7,1);

end

testMusic = abs(testMusic);

testMusic = testMusic - mean(testMusic);

transformedMusic = (Transform \* testMusic)';

musicPredictions = nb.predict(transformedMusic);

correctBayes = correctBayes + sum(musicPredictions=="music");

% repeat for Babble

[m,Fs] = audioread('E:\FakeDesktop\voices\VOiCES\_devkit\distant-16k\distractors\rm3\babb\Lab41-SRI-VOiCES-rm3-babb-mc01-stu-clo.wav');

testBabble = zeros(8193\*7, 20);

for j=1:20

testBabble(:,j) = reshape(spectrogram(m(j\*Fs\*30:2:j\*Fs\*30+Fs\*5+1,1)),8193\*7,1);

end

testBabble = abs(testBabble);

testBabble = testBabble - mean(testBabble);

transformedBabble = (Transform \* testBabble)';

babblePredictions = nb.predict(transformedBabble);

correctBayes = correctBayes + sum(babblePredictions=="babble");

% repeat for TV

[m,Fs] = audioread('E:\FakeDesktop\voices\VOiCES\_devkit\distant-16k\distractors\rm3\tele\Lab41-SRI-VOiCES-rm3-tele-mc01-stu-clo.wav');

testTv = zeros(8193\*7, 20);

for j=1:20

testTv(:,j) = reshape(spectrogram(m(j\*Fs\*30:2:j\*Fs\*30+Fs\*5+1,1)),8193\*7,1);

end

testTv = abs(testTv);

testTv = testTv - mean(testTv);

transformedTv = (Transform \* testTv)';

tvPredictions = nb.predict(transformedTv);

correctBayes = correctBayes + sum(tvPredictions=="tv");

% repeat for voice

directories = dir('E:\FakeDesktop\voices\VOiCES\_devkit\distant-16k\speech\test\rm1\none');

testVoice = zeros(8193\*7, 20);

index = 1;

for i=1:length(directories)

if directories(i).name == "." | directories(i).name == ".."

continue

end

%directories(i).name

voices = dir(strcat(directories(i).folder, '\', directories(i).name));

for j=1:2:length(voices)

if voices(j).name == "." | voices(j).name == ".."

continue

end

[m,Fs] = audioread(strcat(voices(j).folder, '\', voices(j).name));

testVoice(:,index) = reshape(spectrogram(m(Fs\*1:2:Fs\*6+1,1)),8193\*7,1);

index = index + 1;

if index == 21

break;

end

end

if index == 21

break;

end

end

testVoice = abs(testVoice);

testVoice = testVoice - mean(testVoice);

transformedVoice = (Transform \* testVoice)';

voicePredictions = nb.predict(transformedVoice);

correctBayes = correctBayes + sum(voicePredictions=="voice");

Accuracy = correctBayes / 80