Devin Trejo

ECE 3522: Stochastic Processes in Signals and Systems

Department of Electrical and Computer Engineering, Temple University, Philadelphia, PA 1912

I. PROBLEM STATEMENT

Class assignment four has us exploring the distribution fits laid out in chapter four and seeing how they fit real world data sets. We again use our speech signal and google stock as a basis for our analyses. From chapter four we are introduced to a number of

Bernoulli **Binomial** **Geometric**

Exponential **Normal (Gaussian)** **Uniform**

These different distribution models help us model random variables in things like our speech signal or google stock. What we hope to accomplish in this write up is to see how we can use these distribution models to predict an expected outcome.

II. APPROACH AND RESULTS

To begin the assignment we take our speech signal and find its histogram. Once we have a histogram (or pdf) of our speech signal, a good first approach to try a normal distribution. We lay our distribution on top of our histogram to see the fit.

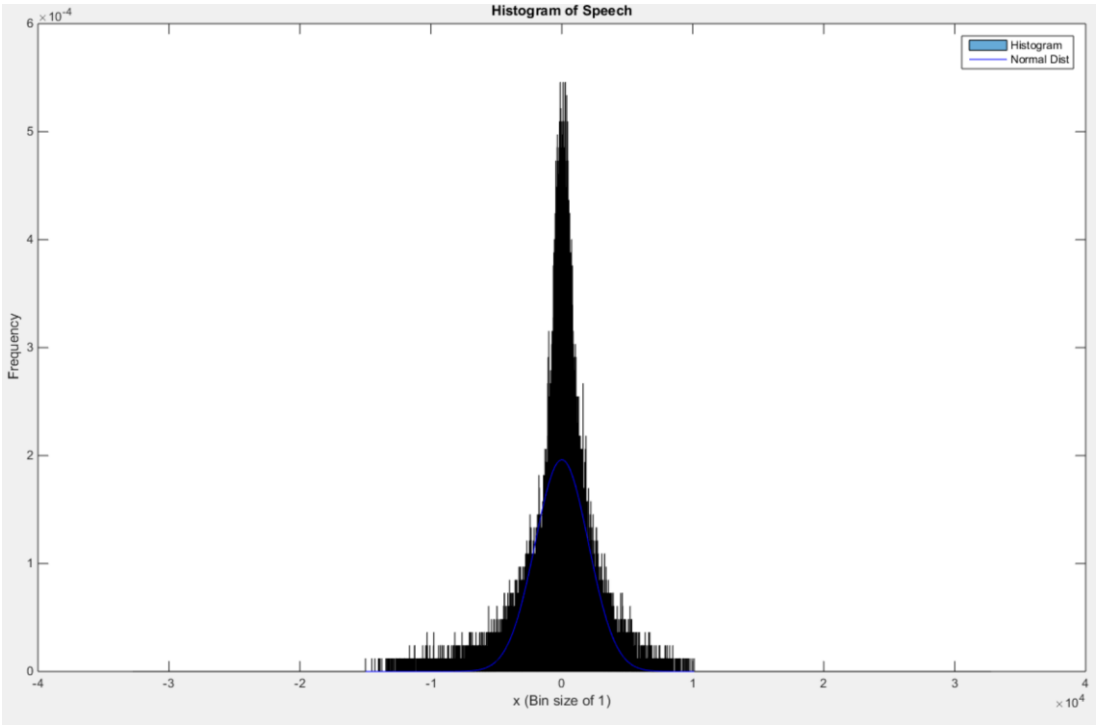

Figure 1: A histogram plot of the speech signal also showing a normal distribution fit.

Due to the possible ranges of our audio signal [-32768:32768] there are many samples whose values become outliers when compared to the mean/variance. Therefore, one may think the correlation between to be low. Yet, our assumption of trying a normal distribution first pays off. With the way our audio signal is constructed we see how

most of our signal is centered around zero. A normal distribution is also centered around the mean making the correlation between the two high. In our analysis we compared the histogram to normal distribution fit to find a mean squared error between the two.

```
MSE_fnNormal =

    0.0021
```

Although our audio signal may have samples whose values are outliers we see a low mean squared error. If our distribution matched our histogram perfectly our mean square error would be zero. If the two were uncorrelated our mean squared error would approach infinity. Again since our distribution fits nicely our correlation resulted in a value of 0.0021. Therefore we can use our distribution model to represent pretty accurately the characteristics of our audio signal.

Our distribution does have problems fitting to the peaks in frequency that occur in our audio signal around zero. As is seen in Figure 2 our distribution does not predict frequencies 5.5E-4 but instead a maximum frequency of ~2E-4. It does accurately predict our frequencies of our outliers for values less than -6000 and greater than 6000. Our samples in these ranges rarely occur. These characteristics are what cause our mean square error to be relatively low.

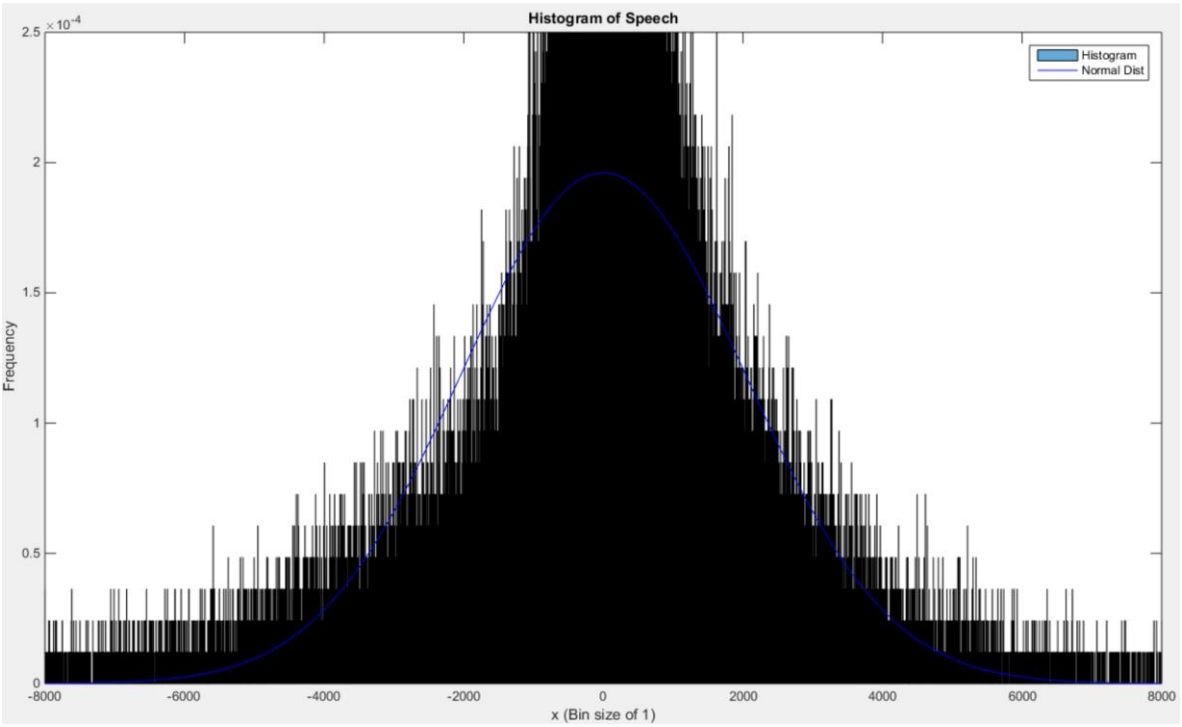

Figure 2: A histogram plot of the speech signal also showing a normal distribution fit (Zoomed in to sample value ranging: [-6000:6000])

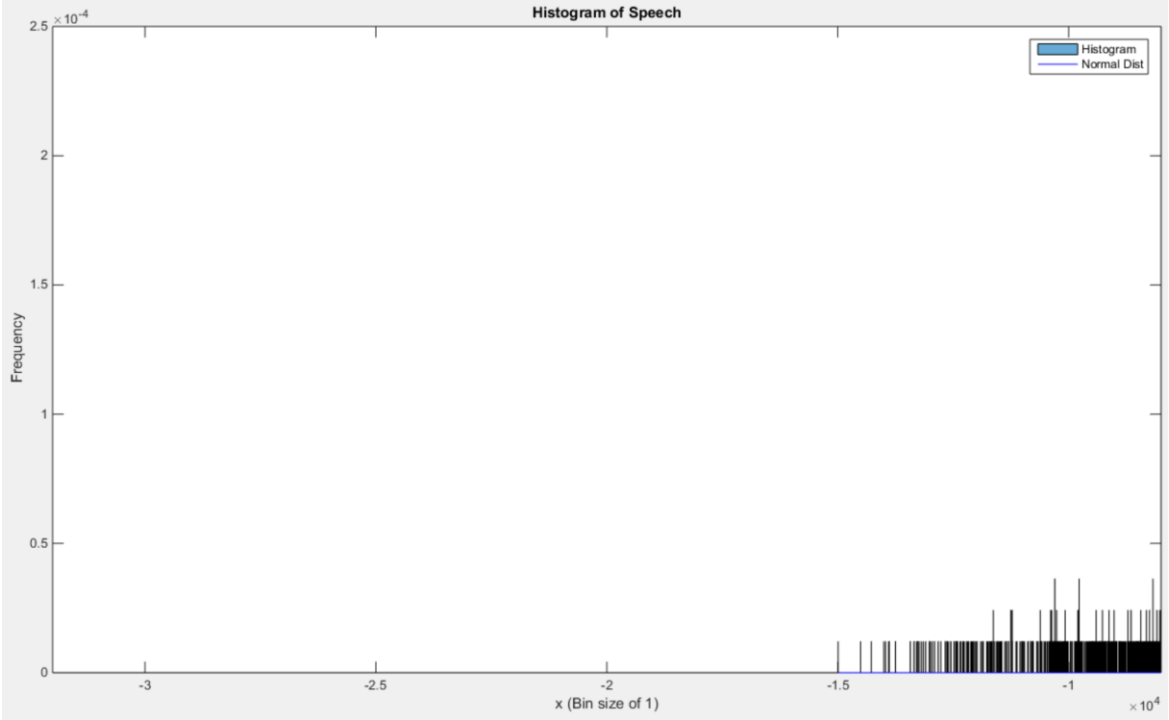

Figure 3: A histogram plot of the speech signal also showing a normal distribution fit (Zoomed in to sample values ranging: [0:-6000])

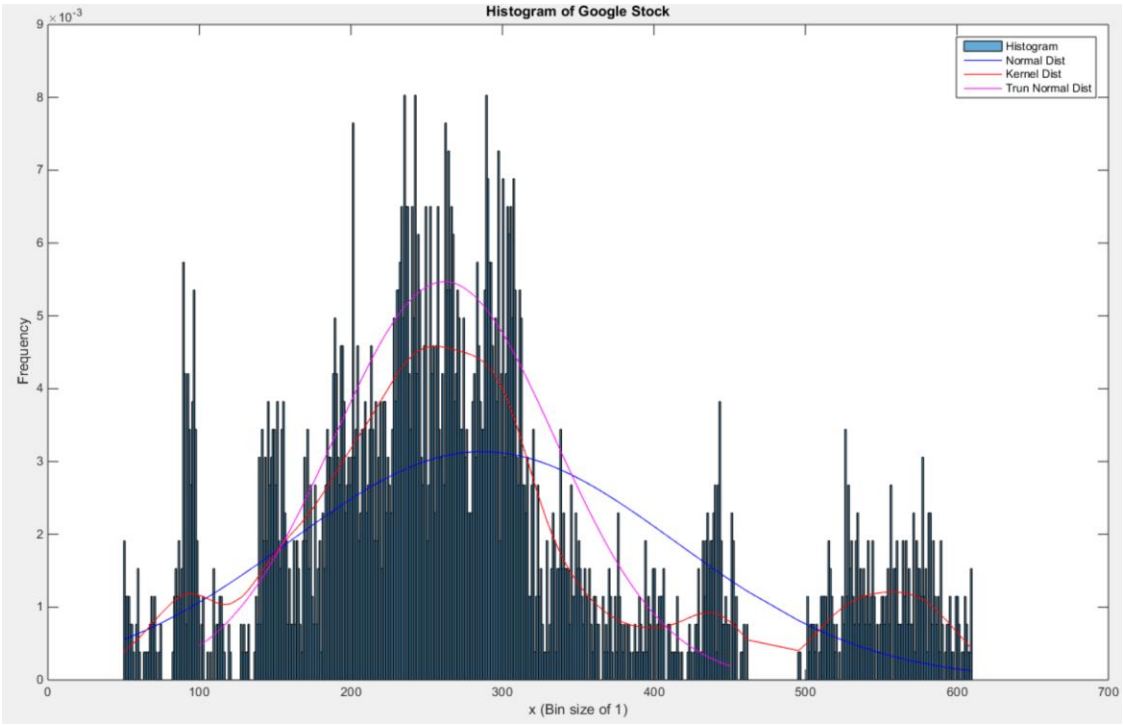

Figure 4: A histogram plot of the google stock also showing three distributions fits.

Our second data set does not fit nicely into a normal distribution. From our earlier computations we know the mean for the close google stock is 286 with a variance of 16194. For comparison our speech signal had a mean of -0.389

with a variance of 4139362. Using these values we see how the google stock will have a more clustered together histogram which does not fit into the normal distribution well. Performing the same mean square error calculation between our google stock histogram and a normal distribution fit gives:

```
MSE_googleNormal =

    0.0098
```

Comparing the mean squared error for the google close data to our audio signal mean squared error we see a rather sharp increase. Using other distribution model we can better model our data. At first a truncated normal distribution seemed logical. In fact it fits pretty well given you take your data to range from 100 to 450. However this distributed model only works for this range of data. How can we fit a distribution to the entire set?

After some research we found a kernel distribution. The kernel distribution is a weighted sum of normal distributions. We already know a truncated normal fits the data rather well if we restrict its range. Fitting the kernel to the histogram we see a better fit.

```
MSE_googleKernel =

    0.0081
```

Our mean squared error drops by 18 points, a better correlation!

III. MATLAB CODE

```
%%
clear; clc; close all;

% Let's first open the raw speech data file and store its values in a
% vector fn
%
fp=fopen('rec_01_speech.raw', 'r');
% Test Sine Wave
    %fp=fopen('rec_01_sine.raw', 'r');
fn=fread(fp,inf,'int16');
fclose(fp);

L_speech = length(fn);
% We are given a sample frequency of 8 kHz
%
fs = 8000;
L_speech = length(fn);
timeL = L_speech/fs;

% We can find the legnth of our signal given our sample frequency
%
t= linspace(0, timeL,L_speech);

% Let's open the xls data file and store its values in avector fn
%
google_v00 = xlsread('google_v00.xlsx');
% google_open = google_v00(:,1);
% google_high = google_v00(:,2);
% google_low = google_v00(:,3);
google_close = google_v00(:,4);
L_googleClose = length(google_close);

clear google_v00

% Let us find the min/max val, mean, median, and variance
%
google_min = min(google_close);
google_max = max(google_close);
google_mean = mean(google_close);
google_median = median(google_close);
google_var = var(google_close);

fn_min = min(fn);
fn_max = max(fn);
fn_mean = mean(fn);
fn_median = median(fn);
fn_var = var(fn);

% Print our findings
%

out = sprintf('Google data: min = %f, max = %f, mean = %f, median = %f, variance = %f
\n'...
    , google_min, google

out = sprintf('Speech data: min = %f, max = %f, mean = %f, median = %f, variance = %f
\n'...
    , fn_min, fn_max, fn_mean, fn_median, fn_var);
disp(out);
```

1 To begin we load in our signal (the same method used in previous assignments). Once we have a data organized into variables inside MATLAB we find the mean and variance of the Google Stock price and speech signal using MATLAB's built in functions.

```
bin_size = 1;

% ------------------ Speech ------------------

% Compute Histogram and plot histogram
%
bounds = [-32768 round(fn_min/5)*5:bin_size:round(fn_max/5)*5 32768];
figure('name','[ECE 3522] Class Assignment [2]');
h_fn = histogram(fn, bounds, 'Normalization', 'probability');
title('Histogram of Speech');
xlabel(sprintf('x (Bin size of %d)',bin_size));
ylabel('Frequency');
hold on
```

2 Using the histogram function we find our histogram (PDF). We used the function previously in other class assignments. The bounds we set are the entire range for a 32 bit integer (32768). We also hold on the plot so we can plot the distribution functions later on top of the histogram.

```
% Sort the Speech Data
fn_sort = sort(fn);

% Upsample the histogram frequencies
index = 1;
hfnUpSam = zeros(L_speech, 1);
hist_fnValue = h_fn.Values;

for D = floor(fn_min):bin_size:floor(fn_max)-1
    for k = 1:L_speech
        if ((D <= fn(k)) && (fn(k) < D+1))
            hfnUpSam(k) = hist_fnValue(index);
        end
    end
    index = index + 1;
end
```

3 After we find our histogram we need to up-sample the values to be the same size as our speech signal. Our histogram is smaller than our speech signal now since we group our histogram values into bin sizes of 1. To up-sample we loop through every value possible in our speech signal (D) and compare it to values in our speech signal (fn(k)). If D matches a value in our signal we store the frequency produced by the histogram function in a new array hfnUpSam.

```
% Use a function to fit the data to a Normal distribution
pd = fitdist(fn_sort, 'Normal');

% Find values of the PDF for our data
yfnNormal = pdf(pd,fn_sort);

% Plot the Normal Distribution
plot(fn_sort, yfnNormal, 'b');

% Label the axis
legend('Histogram', 'Normal Dist');
```

4 Next we find the distribution function. We first specify we want a Normal distribution. The pdf function fits our data to the normal distribution. After we have pdf we just need to plot the distribution. The pdf will be of same length as our original signal.

```
% Find MSE for the Normal on Speech Signal
for k = 1:L_speech
    xt(k) = (yfnNormal(k)-hfnUpSam(k))^2;
end

MSE_fnNormal = sum(xt)

hold off
```

5 After we have our pdf and up-sampled we can find the mean squared error using the following form:

$$MSE = \sum (PDF - HISTOGRAM)^2$$

```
% ------------------ Google ------------------
clear xt pd

% Compute Histogram and plot histogram
%
bounds = [round(google_min/5)*5:bin_size:round(google_max/5)*5];
figure('name','[ECE 3522] Class Assignment [2]');
h_google = histogram(google_close, bounds, 'Normalization', 'probability');
title('Histogram of Google Stock');
xlabel(sprintf('x (Bin size of %d)',bin_size));
ylabel('Frequency');
hold on

% Upsample the histogram frequencies
index = 1;
hGoogleUpSam = zeros(L_googleClose, 1);
hist_googleValue = h_google.Values;

for D = floor(google_min):bin_size:floor(google_max)-1
    for k = 1:L_googleClose
        if (D <= google_close(k) && google_close(k) < D+1)
            hGoogleUpSam(k) = hist_googleValue(index);
        end
    end
    index = index + 1;
end

% Google Normal Distribution Computation
%

% Sort the Google Data
google_close_sort = sort(google_close);

% Use a function to fit the data to a Normal distribution
pd = fitdist(google_close_sort, 'Normal');

% Find values of the PDF for our data
yGoogleNormal = pdf(pd,google_close_sort);

% Plot the Normal Distribution
plot(google_close_sort, yGoogleNormal, 'b');

% Find MSE for the Normal on GoogleClose
for k = 1:L_googleClose
    xt(k) = (yGoogleNormal(k)-hGoogleUpSam(k))^2;
end

MSE_googleNormal = sum(xt)
```

6 We repeat the steps taken with the signal with audio signal but now using our google closing stock data.

```
clear xt pd

% Use a function to fit the data to a Kernel distribution
pd = fitdist(google_close, 'Kernel');

% Find values of the PDF for our data
yGoogleKernel = pdf(pd,google_close_sort);

% Plot the Kernel Distribution
plot(google_close_sort, yGoogleKernel, 'r');

% Find MSE for the Kernel on GoogleClose
for k = 1:L_googleClose
    xt(k) = (yGoogleKernel(k)-hGoogleUpSam(k))^2;
end

MSE_googleKernel = sum(xt)
```

7 For the google stock we noted that a normal distribution did not fit the data too well. To fit a Kernel distribution to the data we need to pass the pdf function a new handle. Using the fitdist function we tell it to fit a kernel distribution to our data. After we can find plot and find the MSE as we did using the process as our normal distribution.

```
% Find the truncated Normal Distribution
for i = 1:length(google_close_sort)
    if (google_close_sort(i)) <= 100
        indexMin = i;
    end
    if (google_close_sort(i)) >= 450
        indexMax = i;
        break
    end
end

% Find the new variance and Means for the truncated GoogleClose
google_trun_var =  var(google_close_sort(indexMin:indexMax));
google_trun_mean = mean(google_close_sort(indexMin:indexMax));

% Use a function to fit the data to a Normal distribution
pd = makedist('Normal', 'mu',google_trun_mean,'sigma',sqrt(google_trun_var));

yGoogleTrunNormal = pdf(pd,google_close_sort(indexMin:indexMax));
plot(google_close_sort(indexMin:indexMax), yGoogleTrunNormal, 'm');
legend('Histogram', 'Normal Dist', 'Kernel Dist', 'Trun Normal Dist');
```

8 For the truncated normal we use the same function. However, instead of passing the pdf function the entire signal we pass it a filtered version of our google stock. Using a ‘for loop’ we loop through a sorted version of our google stock data. If the stock data is between 100 we store that index of the sorted google stock into a variable. The same is down for the upper bound index. We re-compute the variance and mean so that we can make our normal distribution.

IV. CONCLUSIONS

The conclusion for this lab is that we can model sets of random variable data using known distribution function. Finding a distribution that fits can be tricky but a normal distribution can give you a good starting point. Knowing you have a good distribution can then be used to model random variable data to a degree.

An actual application is when finding the rate of defection for products coming off an assembly line. You can feed your data into a distribution model and predict the rate of failing products. Companies will find this useful when allocating a budget for servicing these defective products.