BE 521: Homework 3 Questions

Feature extraction

Spring 2020

68 points

Due: Thursday, 2/13/2020 11:59pm

Objective: Extract features from data and build a simple detector

Andrew Clark
Collaborators: Joe Iwaysk, Vishal Tien, Alex Silva

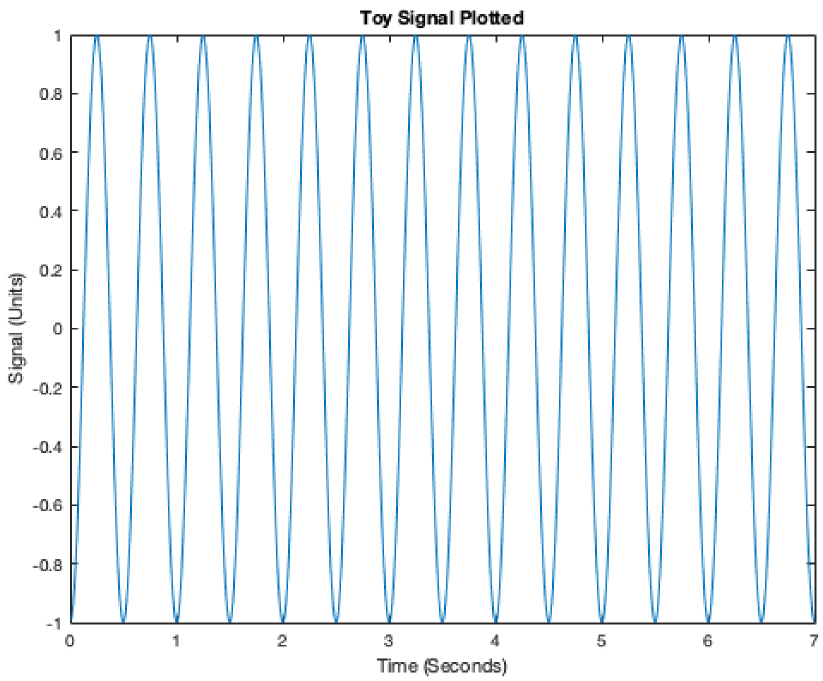
1 Features and Simulations (39 pts)

As you learned in class, features are the backbone of almost all detection strategies, from seizures in EEG to faces in images. Features are usually defined in journal articles as an equation or set of equations, and the task for the reader—if she wants to use that feature—is to implement that feature in code. In this section, you will explore and implement some features commonly used in EEG analysis and test them on simulated time-series data.

1. Consider the following toy signal: 7 seconds of a 2 Hz sine wave with a quarter period phase-shift, sampled at 100 Hz

   (a) Plot the signal. (2 pts)

```
%create sine wave
freq=2;
samp_freq=100;
dur=7;
time=1/samp_freq:(1/samp_freq):dur;

output=sin(2*freq*pi*time-(pi/2));

%create plot
figure
plot(time,output)
xlabel('Time (Seconds)')
ylabel('Signal (Units)')
title('Toy Signal Plotted')
```



(b) Using the Matlab functions for the difference, sum, and absolute value of the elements in a vector (look them up if you don't know them), create an anonymous function for the line-length feature $LL(\mathbf{x}) = \sum_{i=2}^{n} |x_i - x_{i-1}|$ in one line of Matlab code that uses no loops (i.e., the outputs of one function will be the inputs of another). Your function should look something like

```
      LLFn = @(x) XXXXXX;
```

where XXXXX represents some use of the aformentioned functions and the input signal x. (4 pts)

```
%define line length function
LLfn = @(x) sum(abs(diff(x))); %use the diff function to calculate the difference between elements
```

(c) What is the line length of this signal? (2 pts)

```
LLfn(output)
```

```
ans =

   55.9921
```

As calculated by the code above, the line length of this signal is 55.99 uV.

2. Consider line length of the signal using a sliding window with a certain amount of window overlap (or, to think of it another way, displacement with each "slide"). Now, instead of having just one value for the line length, you will have a number of values.

(a) Given a signal **x** with sampling frequency **fs** and windows of length **winLen** and displacement **winDisp** (both in seconds), create an anonymous function called **NumWins** that calculates the number of possible (full) windows in your signal of length **xLen** (in samples), i.e.,

```
      NumWins = @(xLen, fs, winLen, winDisp) XXXXXX;
```

where **XXXXXX** is the single-line expression for this value. You may assume that **winDisp** is a factor of both **winLen** (as it usually is/should be) and the length (in seconds) of **x**. (4 pts)

```
%This function calculates how many possible windows there are in your
%signal;
NumWins = @(xLen, fs, winLen, winDisp) floor((((xLen/fs)-winLen)/winDisp)+1);

%xLen/fs = length of signal in seconds

%xLen in samples
```

(b) Use this function to calculate the number of windows for the signal described in Question 1.1 for a 500 ms window with 250 ms displacement, i.e., the expression

```
      NumWins(length(x), fs, winLen, winDisp)
```

where **fs**, **winLen**, and **winDisp** are the appropriate values. (1 pt)

```
x=output;
NumWins(length(x), 100, 0.5, 0.25)
```

```
ans =

    27
```

As calculated in the code above the number of windows is equal to 27.

(c) Repeat the above calculation for 50 ms window displacement. (1 pt)

```
x=output;
NumWins(length(x), 100, 0.5, 0.05)
```

```
ans =

   131
```

As calculated in the code above the number of windows is equal to 131.

(d) Repeat the above calculation for 100 ms window displacement. (1 pt)

```
x=output;
NumWins(length(x), 100, 0.5, 0.1)
```

```
ans =
66
```

As calculated in the code above the number of windows is equal to 66.

3. (a) Create a function (in another file) called `MovingWinFeats(x, fs, winLen, winDisp, featFn)` that returns a vector of the values of the feature on the signal `x` in all the possible windows, where `featFn` is a feature function like the one you wrote in Question 1.1.b. You may find it useful to use your `NumWins` function (or at least its expression). You may assume that the product of `winDisp` and the sampling rate `fs` is an integer. (6 pts)

   Make sure your MovingWinFeats code is in your pdf. One way is to use the following Matlab code (in your script) to automatically load in the function's code (where we assume that the function is one directory up from the *.tex file). Windows users may need to change the forward-slash to a backslash.

```
%   <latex>
%   \lstinputlisting{[path to] MovingWinFeats.m}
%   </latex>
```

```
function [features] = MovingWinFeats(x,fs,winLen,winDisp,featFn)
%Function that returns a vector of the values of the feature on the signal x in all possible windows
%Calculates line length in every window

NumWindows=floor((((length(x)/fs)-winLen)/winDisp)+1);
features=zeros(1,NumWindows);

%calculate line length in each window
starter=1;
for i=1:NumWindows
    %set up x
    in_between=x(starter:starter+winLen*fs-1);
    features(i)=featFn(in_between);
    starter=starter+winDisp*fs;
end

end
```

(b) Using the signal you defined in Question 1.1 and the function you created in Question 1.1.b, calculate the line-length over windows of length 500 ms and displacement 250 ms. (2 pts)

```
%This line gives the desired vector
ans=MovingWinFeats(output,100,0.5,0.25,LLfn)
```

```
ans =

  Columns 1 through 7

    3.9921    3.9921    3.9921    3.9921    3.9921    3.9921    3.9921

  Columns 8 through 14

    3.9921    3.9921    3.9921    3.9921    3.9921    3.9921    3.9921

  Columns 15 through 21
```

```
    3.9921    3.9921    3.9921    3.9921    3.9921    3.9921    3.9921

  Columns 22 through 27

    3.9921    3.9921    3.9921    3.9921    3.9921    3.9921
```

(c) Add a unit-amplitude 5 Hz signal (in the form of a sine wave) to your original signal and again calculate the line length over the same window and displacement. (2 pts)

```
new_freq=5;
new_signal=output+sin(2*pi*time*new_freq);

%This line gives the desired vector
feat=MovingWinFeats(new_signal,100,0.5,0.25,LLfn)
```

```
feat =

  Columns 1 through 7

    9.6262   10.3231   10.5157   10.3389    9.6262   10.3231   10.5157

  Columns 8 through 14

   10.3389    9.6262   10.3231   10.5157   10.3389    9.6262   10.3231

  Columns 15 through 21

   10.5157   10.3389    9.6262   10.3231   10.5157   10.3389    9.6262

  Columns 22 through 27

   10.3231   10.5157   10.3389    9.6262   10.3231   10.5157
```

4. Code the following 3 additional features in MINIMAL lines of code (hint: each can be implemented in one line using the anonymous function trick).

(a) Area, $A(\mathbf{x}) = \sum_{i=1}^{n} |x_i|$ (2 pts)

```
%Area feature function
A = @(x) sum(abs(x));
```

(b) Energy, $E(\mathbf{x}) = \sum_{i=1}^{n} x_i^2$ (2 pts)

```
%Energy feature function
E = @(x) sum(x.^2);
```

(c) Zero-Crossings around mean,

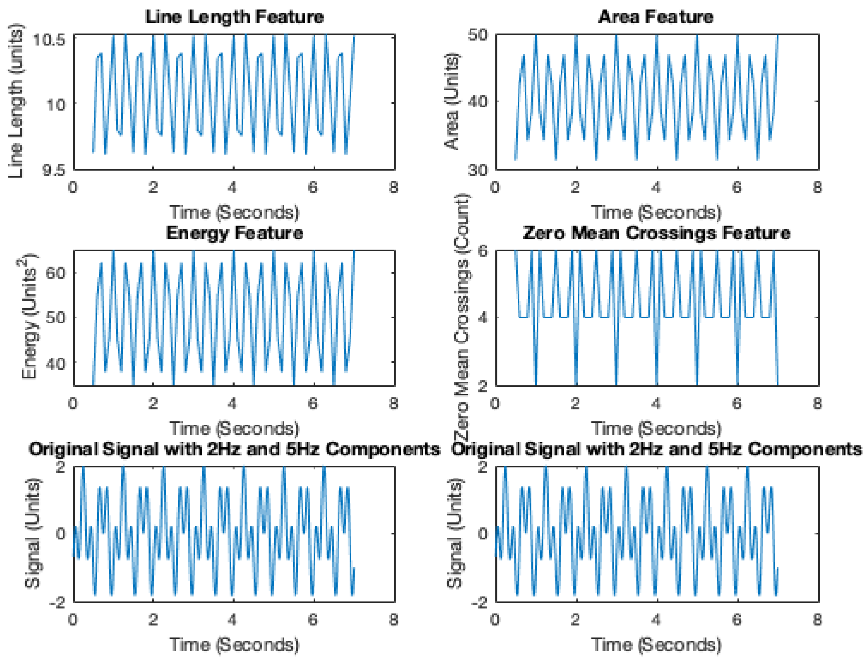
$ZX(\mathbf{x}) = \sum_{i=2}^{n} \mathbf{1}(\textbf{FromAbove})$ OR $\mathbf{1}(\textbf{FromBelow})$, where $\mathbf{1}(\cdot)$ denotes the indicator function, which returns a zero if its argument is false and a one if it is true, **FromAbove** denotes $(x_{i-1} - \overline{x} > 0)$ AND $(x_i - \overline{x} < 0)$, **FromBelow** denotes $(x_{i-1} - \overline{x} < 0)$ AND $(x_i - \overline{x} > 0)$, and $\overline{x}$ is the mean value of the elements in x. (4 pts)

```
%Zero—Crossings around mean function
ZX = @(x) sum(abs(diff((x—mean(x))>0)));
```

(d) Plot the values of the four features on the combined signal in the first four cells of a 3x2 matlab subplot. Use a 500 ms window with 100 ms displacement. Using the right-aligned convention (where the "official" time of the feature is that of the last data point in the window), give the appropriate time axis for each window point. In addition, plot the original signal with the 2Hz and 5Hz components in the last two cells of the 3x2 subplot (to make comparing down the column easy). Ensure that the time axis in all of your plots is the same. (6 pts)

```
%calculate features
line_length=MovingWinFeats(new_signal,100,0.5,0.1,LLfn);
area=MovingWinFeats(new_signal,100,0.5,0.1,A);
energy=MovingWinFeats(new_signal,100,0.5,0.1,E);
zero_mean_crossings=MovingWinFeats(new_signal,100,0.5,0.1,ZX);

time2=0.5:0.1:7;
```

```
%create subplots
figure
subplot(3,2,1)
plot(time2,line_length)
xlabel('Time (Seconds)')
ylabel('Line Length (units)')
title('Line Length Feature')
subplot(3,2,2)
plot(time2,area)
xlabel('Time (Seconds)')
ylabel('Area (Units)')
title('Area Feature')
subplot(3,2,3)
plot(time2,energy)
title('Energy Feature')
ylabel('Energy (Units^2)')
xlabel('Time (Seconds)')
subplot(3,2,4)
plot(time2,zero_mean_crossings)
xlabel('Time (Seconds)')
ylabel('Zero Mean Crossings (Count)')
title('Zero Mean Crossings Feature')
subplot(3,2,5)
plot(time,new_signal)
xlabel('Time (Seconds)')
ylabel('Signal (Units)')
title('Original Signal with 2Hz and 5Hz Components')
subplot(3,2,6)
plot(time,new_signal)
xlabel('Time (Seconds)')
title('Original Signal with 2Hz and 5Hz Components')
ylabel('Signal (Units)')
```



2 Feature Overlays (17 pts)

In this section, you will use a line-length feature overlay on a segment of EEG containing a seizure. This data is stored in I521_A0003_D001

1. What is the length using hours:minutes:seconds:milliseconds of the recording? (Use getDuration) (2 pts)

```
session = IEEGSession('I521_A0003_D001', 'andrewc', '/Users/andrewclark/Downloads/ieeg_password.bin' );
```

```
Warning: Objects of edu/upenn/cis/db/mefview/services/TimeSeriesDetails class
exist — not clearing java
Warning: Objects of edu/upenn/cis/db/mefview/services/TimeSeriesInterface class
exist — not clearing java
IEEGSETUP: Found log4j on Java classpath.
URL: https://www.ieeg.org/services
Client user: andrewc
Client password: ****
```

```
durationInUSec = session.data(1).rawChannels(1).get_tsdetails.getDuration;
durationInSec = durationInUSec / 1e6;
durationInMin = durationInSec/60;
```

```
sec= seconds(durationInSec);
sec.Format= 'hh:mm:ss.SSS'
```

```
sec =

  duration

   01:21:20.390
```

As shown in the code above the length is 01 hours: 21 minutes: 20 seconds: 390 milliseconds

2. How many data points should we discard at the end if we want to clip the recording to the last full second? Do this clipping. (1 pt)

```
nr = ceil((session.data.rawChannels(1).get_tsdetails.getEndTime)/1e6*session.data.sampleRate);
allData = session.data.getvalues(1:nr,1);
session.data
```

```
ans =

  <a href="matlab:help('IEEGDataset')">IEEGDataset</a>:

         snapName: 'I521_A0003_D001'
          montage: As Recorded
           filter: 'No Filter'
         resample: 'Not resampled'
       sampleRate: 200
           values:
    channelLabels: [1x2 cell]
         annLayer: []
      rawChannels: [1x1 IEEGTimeseries]
      allMontages: [1x44 IEEGMontage]

  <a href="matlab:methods(IEEGDataset)">Methods</a>, <a href="matlab:IEEGObject.openPortalSite()">main.ieeg.
```

```
sample_rate=200; %in Hz
clipped_points=(sample_rate/1000)*(390)
len=length(allData);
clipped_data=allData(1:(end-clipped_points));
```

```
clipped_points =

    78
```

78 data points should be discarded if we want to clip the recording to the last full second. This clipping is performed by the code above and the clipped dataset is stored to a variable called "clipped_data."

3. If we want to overlay a feature trace on the original signal, we have to interpolate that feature (which has been calculated over windows) for each data point of the original signal. One of the simplest methods of doing this is called zero-order interpolation, where we just hold the value constant until we get to the next calculated value. For example, if we had a moving window of 1 second with 1 second displacement, the zero-order interpolated feature vector would have the same value the entire first second, then the same for the entire second second, etc, where each second contains the same number of points as the sampling frequency of the original signal.

   (a) Using the `repmat` and `reshape` functions, create an external function `zoInterp(x, numInterp` that copies each value of `x` `numInterp` times. You can implement this function in one line of code with no loops. Include the code for this function as you did in Question 1.3.a. (2 pts)

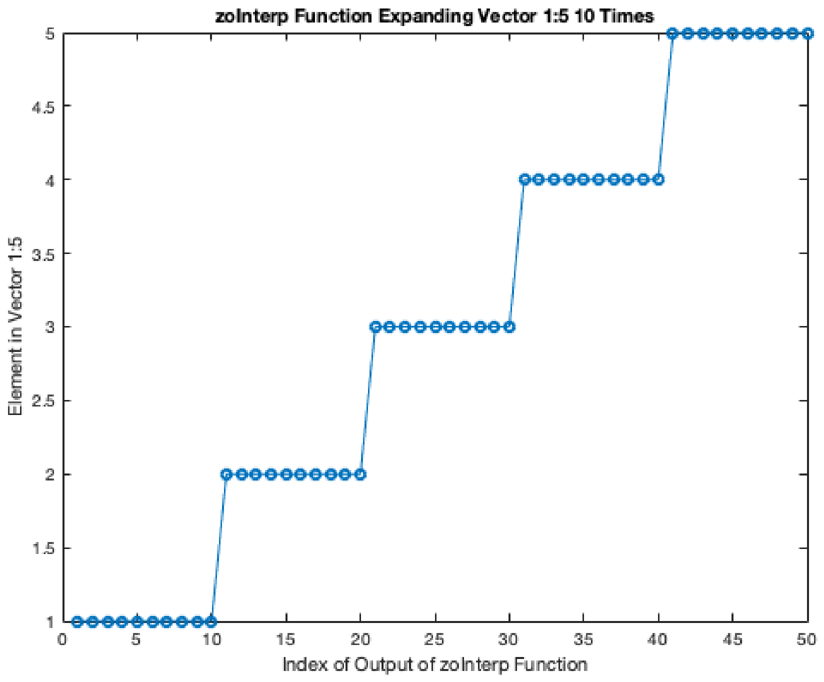
```
%   <latex>
%   \lstinputlisting{zoInterp.m}
%   </latex>
```

(b) Confirm that this function works correctly by expanding the length of the vector `1:5` by a factor of 10 and plotting with the command

```
    plot(zoInterp(1:5,10),'-o')
```

where the `'-o'` option lets us see the individul points as well as the line that connects them. (2 pts)

```
figure
plot(zoInterp(1:5,10),'-o')
title('zoInterp Function Expanding Vector 1:5 10 Times')
ylabel('Element in Vector 1:5')
xlabel('Index of Output of zoInterp Function')
```

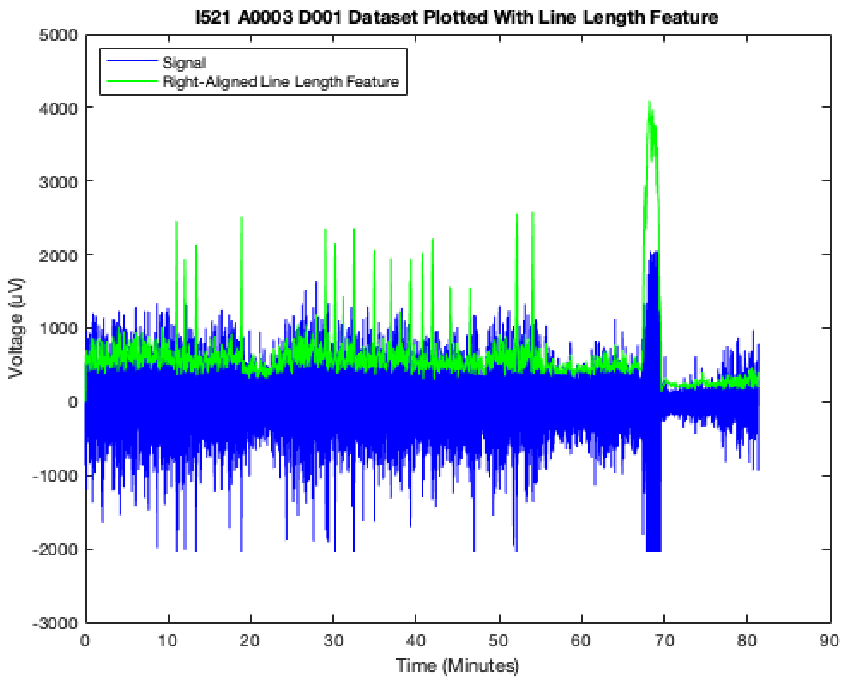


4. Using a 4-second sliding window with 1-second displacement, calculate the line length feature over the entire signal. Normalize the line-length feature values to have a maximum twice that of the original EEG signal maximum. Plot the signal in blue and overlay the right-aligned line-length feature in green. Note: you will need to pad your signal in order to get them to line up correctly and be the same length. Put the units of your time axis in minutes, and be sure to add a legend in a location in the plot that does not cover up any signal or feature. (6 pts)

```
%clip_data_time=durationInSec

line_length_new_sig=MovingWinFeats(clipped_data,sample_rate,4,1,LLfn);
```

```
%plot(line_length_new_sig)
scaling_factor=(2*max(clipped_data))/max(line_length_new_sig);
scaled_length=line_length_new_sig*scaling_factor;
inter_data=zoInterp(scaled_length,floor(length(clipped_data)/length(line_length_new_sig)));
%pad=scaling_factor(end)+zeros(length(

%time vector for the clipped data
%timed=(1/sample_rate):(1/sample_rate):
timed = linspace(1/sample_rate,(length(clipped_data)/sample_rate),length(clipped_data));
timed=timed/60;

pad=zeros(1,length(clipped_data)-length(inter_data));

padded_data_feat=[pad inter_data];

scaled_padded_data=padded_data_feat*scaling_factor;
```

```
figure
plot(timed,clipped_data,'b')
hold on
plot(timed,padded_data_feat,'g')
xlabel('Time (Minutes)')
ylabel('Voltage (uV)')
legend('Signal','Right-Aligned Line Length Feature','Location','northwest')
title('I521 A0003 D001 Dataset Plotted With Line Length Feature')
```

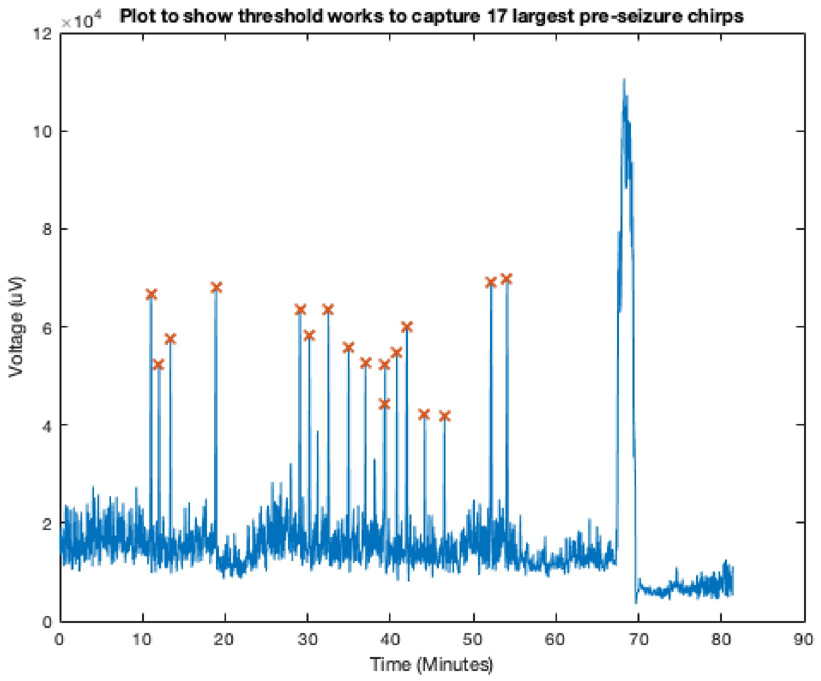


5. What threshold might you use on the raw line-length feature vector (not the normalized one used for plotting) in order to capture the 17 largest pre-seizure chirps that occur? (1 pt)

```
threshold=1550*(1/scaling_factor);
```

```
[pks, locs]=findpeaks(line_length_new_sig, 'MinPeakHeight', threshold);

%make new time vector
time5=4/60:1/60:durationInMin;

plotted_locs=locs(1:17);

plotted_pks=pks(1:17);
```

```
figure
plot(time5,line_length_new_sig)
hold on
plot(time5(plotted_locs),plotted_pks,'x')
xlabel('Time (Minutes)')
ylabel('Voltage (uV)')
title('Plot to show threshold works to capture 17 largest pre-seizure chirps')
```

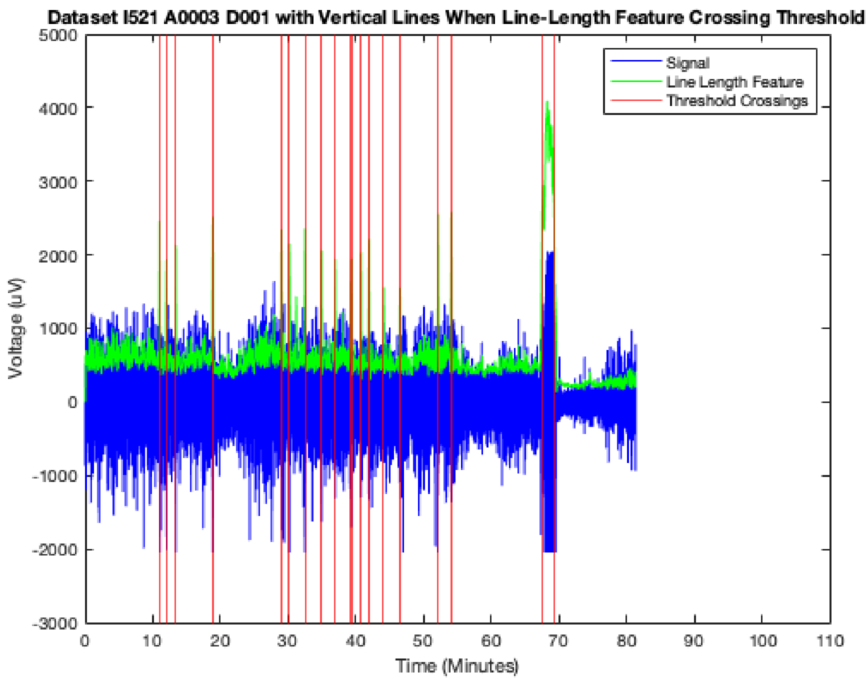


On the raw line-length feature, I would set a threshold of 41,905 uV. The code and figure above just shows that this threshold captures the first 17 peaks of the signal.

6. Using this threshold value, in another plot draw red vertical lines at the leading point in time where the threshold is crossed. Add these vertical lines on top of the plot you made in Question 2.4. These events should capture the pre-seizure chirps, the seizure onset, and some flickering during the end of the seizure. (3 pts)

```
figure
plot(timed,clipped_data,'b')
hold on
plot(timed,padded_data_feat,'g')
for k=2:length(padded_data_feat)
    if padded_data_feat(k)> 1550 && padded_data_feat(k-1) < 1550
```

```
        xline(timed(k),'r','LineWidth',0.75);

    end
end


xlabel('Time (Minutes)')
ylabel('Voltage (uV)')
legend('Signal','Line Length Feature','Threshold Crossings','Location','northeast')
title('Dataset I521 A0003 D001 with Vertical Lines When Line—Length Feature Crossing Threshold')
xlim([0,110])
```



3 Building a Detector (12 pts)

In this section, you will use the features you defined previously to build a seizure detector. Use the EEG data in the file `I521_A0003_D002` with channels `multiSz_1`, and `multiSz_2`.

1. Plot the signal in `multiSz_1` and draw vertical red lines at the times when you think the two seizures begin. (You should be able to do this without the need of any features.) (2 pts)

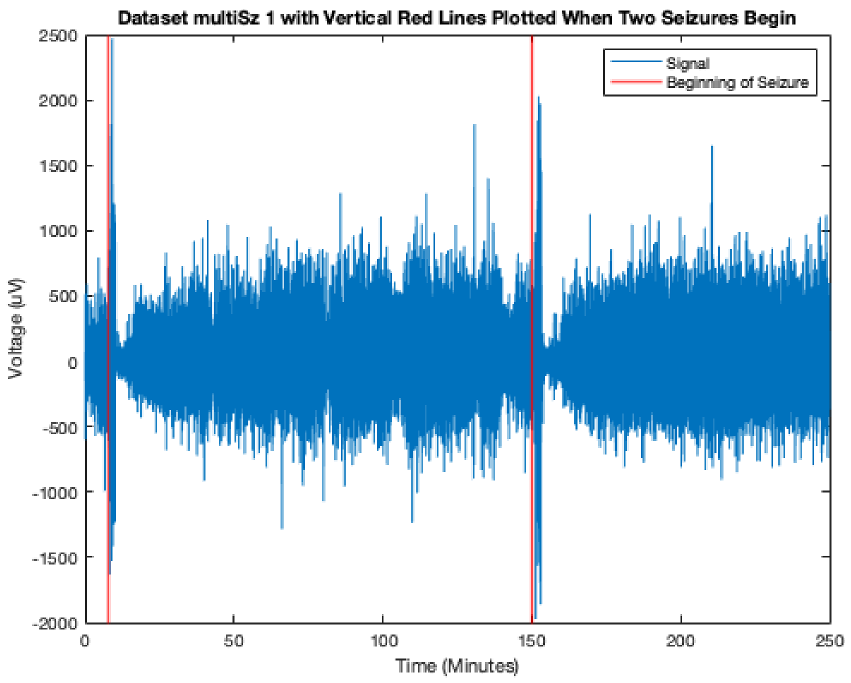
```
session2 = IEEGSession('I521_A0003_D002', 'andrewc', '/Users/andrewclark/Downloads/ieeg_password.bin' );

nr = ceil((session2.data.rawChannels(1).get_tsdetails.getEndTime)/1e6*session2.data.sampleRate);
allData2 = session2.data.getvalues(1:nr,1:2);
session2.data
```

```
Warning: Objects of edu/upenn/cis/db/mefview/services/TimeSeriesDetails class
exist — not clearing java
Warning: Objects of edu/upenn/cis/db/mefview/services/TimeSeriesInterface class
```

```
exist — not clearing java
IEEGSETUP: Found log4j on Java classpath.
URL: https://www.ieeg.org/services
Client user: andrewc
Client password: ****

ans =

  <a href="matlab:help('IEEGDataset')">IEEGDataset</a>:

         snapName: 'I521_A0003_D002'
          montage: As Recorded
           filter: 'No Filter'
         resample: 'Not resampled'
       sampleRate: 200
           values:
    channelLabels: [2x2 cell]
         annLayer: []
      rawChannels: [1x2 IEEGTimeseries]
      allMontages: [1x44 IEEGMontage]

  <a href="matlab:methods(IEEGDataset)">Methods</a>, <a href="matlab:IEEGObject.openPortalSite()">main.ieeg.
```

```
duration_multiSz_1_uS = session2.data(1).rawChannels(1).get_tsdetails.getDuration;
duration_multiSz_2_uS = session2.data(1).rawChannels(2).get_tsdetails.getDuration;
duration_multiSz_1_S=duration_multiSz_1_uS/1e6;
duration_multiSz_2_S=duration_multiSz_2_uS/1e6;

n_sample_rate=session2.data.sampleRate;
```

```
time6=1/n_sample_rate:1/n_sample_rate:duration_multiSz_1_S+1/n_sample_rate;

%time_sz_1_min=time6./60; % convert to minutes

multiSz_1=allData2(:,1);
multiSz_2=allData2(:,2);

figure
plot(time_sz_1_min,multiSz_1)
hold on
xline(7.6,'r','LineWidth',1.5);
xline(150.2,'r','LineWidth',1.5);
%xline(490,'r','LineWidth',1.5);
%xline(9022,'r','LineWidth',1.5);
xlabel('Time (Minutes)')
title('Dataset multiSz 1 with Vertical Red Lines Plotted When Two Seizures Begin')
legend('Signal','Beginning of Seizure','Location','northeast')
ylabel('Voltage (uV)')
```

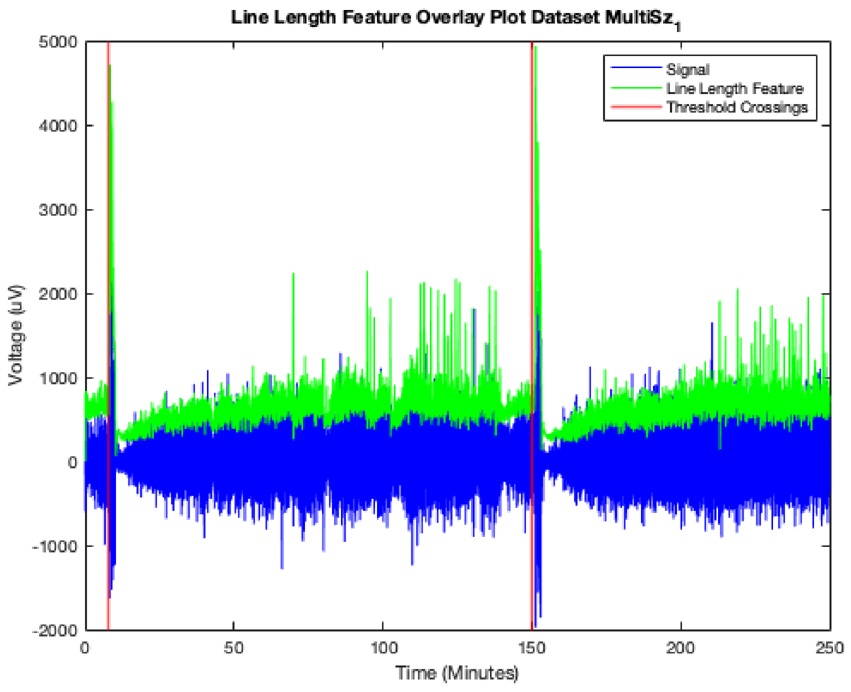


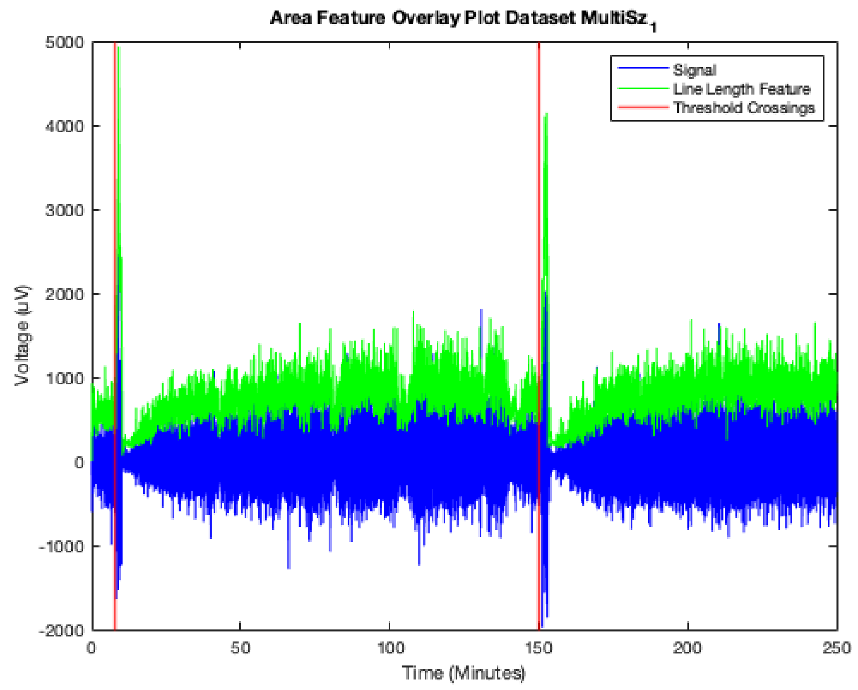
2. Produce feature overlay plots similar to that of Question 2.4 for each of the four features you have implemented along with the red vertical lines at each seizure. Use the same 4-second sliding window with 1 second displacement. (4 pts)

```
%Calculate Features
new_line_length=MovingWinFeats(multiSz_1,n_sample_rate,4,1,LLfn);
new_area=MovingWinFeats(multiSz_1,n_sample_rate,4,1,A);
new_energy=MovingWinFeats(multiSz_1,n_sample_rate,4,1,E);
new_zero_mean_crossings=MovingWinFeats(multiSz_1,n_sample_rate,4,1,ZX);
```

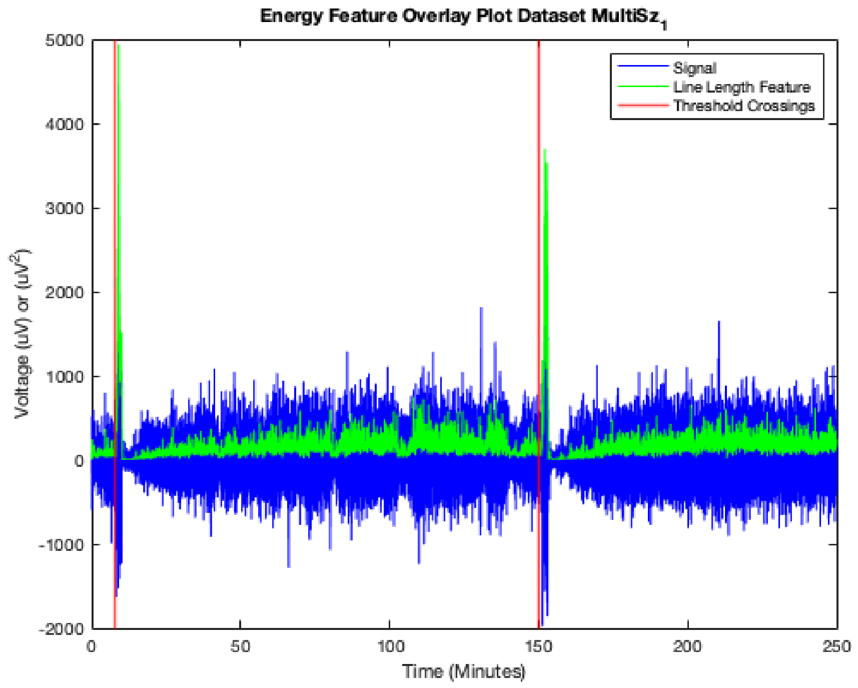
```
%Code to create plot
new_inter_data_length=zoInterp(new_line_length,floor(length(multiSz_1)/length(new_line_length)));
%pad=scaling_factor(end)+zeros(length(

scaling_factor_ll=(2*max(multiSz_1))/max(new_inter_data_length);
new_scaled_ll=new_inter_data_length*scaling_factor_ll;

%time vector for the clipped data
%timed=(1/sample_rate):(1/sample_rate):
timed_new = linspace(1/n_sample_rate,(length(multiSz_1)/n_sample_rate),length(multiSz_1));
timed_new=timed_new/60;

pad_ll=zeros(1,length(multiSz_1)-length(new_scaled_ll));

padded_data_ll_new=[pad_ll new_scaled_ll];

%scaled_padded_data=padded_data_feat*scaling_factor;


%Plot in minutes
figure
plot(timed_new,multiSz_1,'b')
```

```
hold on
plot(timed_new,padded_data_ll_new,'g')
xlabel('Time (Minutes)')
ylabel('Voltage (uV)')
title('Line Length Feature Overlay Plot Dataset MultiSz_1')
xline(7.6,'r','LineWidth',1.5);
xline(150.2,'r','LineWidth',1.5);
legend('Signal','Line Length Feature','Threshold Crossings','Location','northeast')

%Code to create plot for area
new_inter_data_area=zoInterp(new_area,floor(length(multiSz_1)/length(new_area)));
%pad=scaling_factor(end)+zeros(length(

scaling_factor_area=(2*max(multiSz_1))/max(new_inter_data_area);
new_scaled_area=new_inter_data_area*scaling_factor_area;

%time vector for the clipped data
%timed=(1/sample_rate):(1/sample_rate):
timed_new = linspace(1/n_sample_rate,(length(multiSz_1)/n_sample_rate),length(multiSz_1));
timed_new=timed_new/60;

pad_area=zeros(1,length(multiSz_1)-length(new_scaled_area));

padded_data_area_new=[pad_area new_scaled_area];

%Plot in minutes
figure
plot(timed_new,multiSz_1,'b')
hold on
plot(timed_new,padded_data_area_new,'g')
xlabel('Time (Minutes)')
ylabel('Voltage (uV)')
title('Area Feature Overlay Plot Dataset MultiSz_1')
xline(7.6,'r','LineWidth',1.5);
xline(150.2,'r','LineWidth',1.5);
legend('Signal','Line Length Feature','Threshold Crossings','Location','northeast')
%Code to create plot for energy
new_inter_data_energy=zoInterp(new_energy,floor(length(multiSz_1)/length(new_energy)));


scaling_factor_energy=(2*max(multiSz_1))/max(new_inter_data_energy);
new_scaled_energy=new_inter_data_energy*scaling_factor_energy;

timed_new = linspace(1/n_sample_rate,(length(multiSz_1)/n_sample_rate),length(multiSz_1));
timed_new=timed_new/60;

pad_energy=zeros(1,length(multiSz_1)-length(new_scaled_energy));

padded_data_energy_new=[pad_energy new_scaled_energy];

%Plot in minutes
figure
plot(timed_new,multiSz_1,'b')
hold on
plot(timed_new,padded_data_energy_new,'g')
xlabel('Time (Minutes)')
ylabel('Voltage (uV) or (uV^2)')
title('Energy Feature Overlay Plot Dataset MultiSz_1')
xline(7.6,'r','LineWidth',1.5);
xline(150.2,'r','LineWidth',1.5);
legend('Signal','Line Length Feature','Threshold Crossings','Location','northeast')

%Code to create plot for zero crossings around mean
new_inter_data_cross=zoInterp(new_zero_mean_crossings,floor(length(multiSz_1)/length(new_zero_mean_crossings
scaling_factor_cross=(2*max(multiSz_1))/max(new_inter_data_cross);
new_scaled_cross=new_inter_data_cross*scaling_factor_cross;

timed_new = linspace(1/n_sample_rate,(length(multiSz_1)/n_sample_rate),length(multiSz_1));
timed_new=timed_new/60;

pad_cross=zeros(1,length(multiSz_1)-length(new_scaled_cross));

padded_data_cross_new=[pad_cross new_scaled_cross];

%Plot in minutes
figure
plot(timed_new,multiSz_1,'b')
hold on
plot(timed_new,padded_data_cross_new, 'g')
xlabel('Time (Minutes)')
ylabel('Voltage (uV) or Count')
title('Zero Crossings Around Mean Feature Overlay Plot Dataset MultiSz 1')
xline(7.6,'r','LineWidth',1.5);
xline(150.2,'r','LineWidth',1.5);
legend('Signal','Line Length Feature','Threshold Crossings','Location','northeast')
```

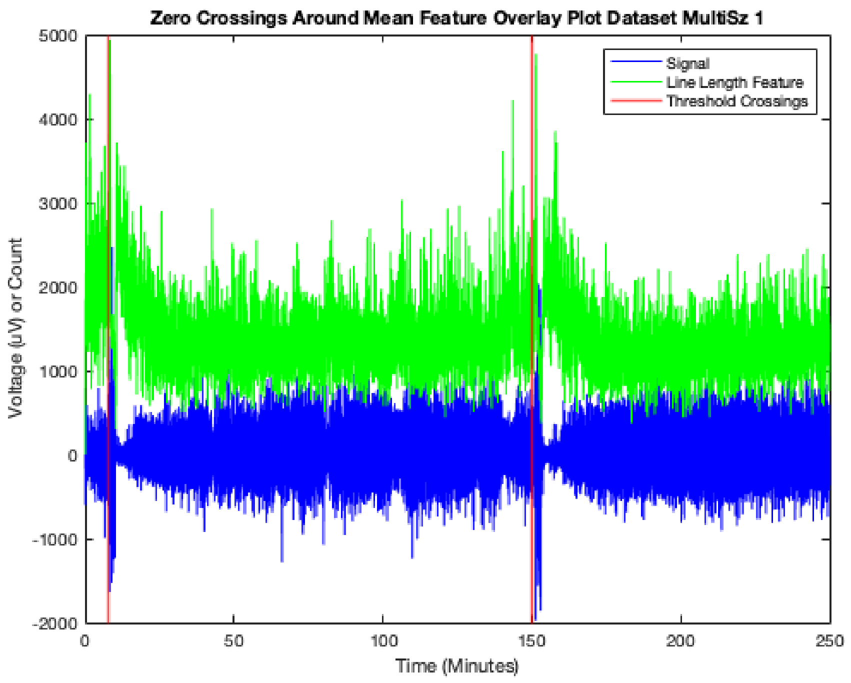




Area Feature Overlay Plot Dataset MultiSz$_1$
Signal
Line Length Feature
Threshold Crossings
Voltage (uV)
Time (Minutes)
5000
4000
3000
2000
1000
0
-1000
-2000
0
50
100
150
200
250



Energy Feature Overlay Plot Dataset MultiSz$_1$
Signal
Line Length Feature
Threshold Crossings
Voltage (uV) or (uV^2)
Time (Minutes)
5000
4000
3000
2000
1000
0
-1000
-2000
0
50
100
150
200
250



3. (a) Based on your plots in the previous question, which of the four features seems to give the largest signal (relative to the background) for when a seizure occurs? Explain why you think this feature is the best. (3 pts)

   Based on the plots created in the previous question, the energy feature gives the largest signal (relative to the background) for when a seizure occurs. This means that energy is the best feature (compared to the other 3 calculated) because the energy feature has the lowest values during non-seizure activity, while still spiking up very high at the same time as seizures in multiSz_1 dataset. So, energy has the lowest signal to noise ratio of the features computed. Thus, energy is the feature that is the most likely to accurately detect seizures and the energy feature will have the lowest-false positive rate.

   (b) What threshold would you use to determine if a seizure is occurring? (1 pt)

```
threshold_seizure=2200*(1/scaling_factor_energy)
```

```
threshold_seizure =

   1.7582e+08
```

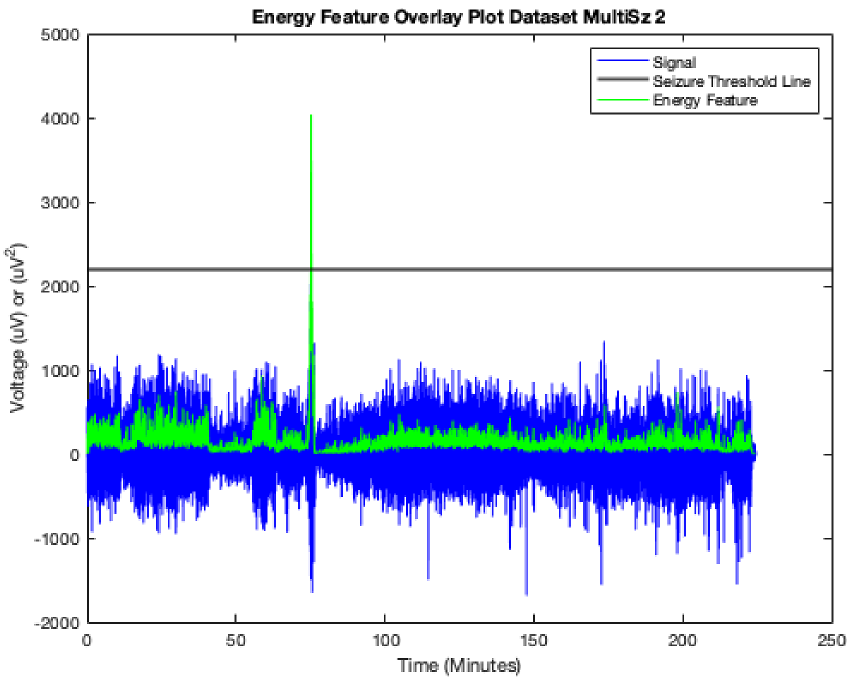
   The threshold that I would use to determine if a seizure is occuring is 1.7582e+08 uV^2. This threshold was chosen by examing the energy feature plot generated in the previous question so that a threshold was chosen to only capture the seizure events and none of the smaller chirps.

4. The signal in **multiSz_2** contains another seizure (whose location should again be fairly obvious). Plot the data along with the feature and threshold (horizontal black line, with correct normalization for the signal in **data2**) you determined in the previous question. (2 pts)

```
time_sz2=time6./60;%convert to minutes
```

```
%calculate energy feature for the multiSz_2 dataset
energy_multiSz2=MovingWinFeats(multiSz_2,n_sample_rate,4,1,E);
```

```
%Code to create plot for energy
inter_data_energy_Sz2=zoInterp(energy_multiSz2,floor(length(multiSz_2)/length(energy_multiSz2)));


scaling_factor_energy_Sz2=(2*max(multiSz_2))/max(inter_data_energy_Sz2);
new_scaled_energy_Sz2=inter_data_energy_Sz2*scaling_factor_energy_Sz2;

timed_new_Sz2 = linspace(1/n_sample_rate,(length(multiSz_2)/n_sample_rate),length(multiSz_2));
timed_new_Sz2=timed_new_Sz2/60;

pad_energy_Sz2=zeros(1,length(multiSz_2)-length(new_scaled_energy_Sz2));

padded_data_energy_new_Sz2=[pad_energy_Sz2 new_scaled_energy_Sz2];

%Plot in minutes
figure
plot(timed_new,multiSz_2,'b')
hold on
yline(threshold_seizure*scaling_factor_energy,'k','LineWidth',2);
plot(timed_new_Sz2,padded_data_energy_new_Sz2,'g')
xlabel('Time (Minutes)')
ylabel('Voltage (uV) or (uV^2)')
title('Energy Feature Overlay Plot Dataset MultiSz 2')
legend('Signal','Seizure Threshold Line','Energy Feature','Location','northeast')
```


Energy Feature Overlay Plot Dataset MultiSz 2
Signal
Seizure Threshold Line
Energy Feature
Voltage (uV) or (uV^2)
Time (Minutes)