BE 521: Homework 6

Spike sorting

Spring 2019

59 points

Due: Thursday, 03/5/2020 11:59pm

Objective: Detect and cluster spikes

Andrew Clark
Collaborators: Alex Silva, Vishal Tien

Overview

In this homework, you will do some basic spike sorting using two different datasets. The first (`I521_A0006_D001`) is from a crayfish neuromuscular junction, a good model for human central nervous system synapses[1]. Specifically, the data contains two simultaneous recordings: an extracellular recording from the third nerve (channel `nerve`) of a crayfish abdominal ganglion, which contains six spontaneously active motor neurons, and an intracellular recording from the superficial flexor muscle (channel `muscle`) innervated by this nerve. You will attempt to discern relationships between the classes of spike waveforms you extract from the motor nerve trace and elicited potentials seen in the muscle fiber recording. Then, you will revisit a human intracranial EEG recording (`I521_A0006_D002`) and use some of the techniques you've learned in class to build a more automated spike sorter. Note: While spikes may have positive and negative deflections, we will only focus on positive spikes on this homework for simplicity.

1 Spike Detection and Clustering (38 pts)

In this section, you will explore some basic filtering and spike thresholding to ultimately compare spike clusters you pick out by eye to those selected by an automated algorithm.

1. You can assume that the nerve samples have already been low-pass filtered. Here you will high-pass filter in order to remove signals like slow local field potentials and 60 Hz power line noise. Create a 4th order *elliptic filter* with 0.1 dB of ripple in the passband, a stopband 40 dB lower than the peak value in the passband, and a passband edge frequency of 300 Hz (see Matlab's `ellip` function and make sure your give the edge frequency in the correct normalized form). The statement to create this filter (defined by the filter coefficients `b` and `a`) should look something like

```
    [b,a]=ellip(n,Rp,Rs,Wp,'high')
```

   Clearly specify the denominator and numerator coefficients obtained for your filter function. (2pts)

[1]The sampling rate of this data is 2000 Hz, which is adequate for this homework's instructional purposes but usually inadequate for real spike sorting, which often uses sampling frequencies on the order of 20 kHz.

```
session = IEEGSession('I521_A0006_D001', 'andrewc', '/Users/andrewclark/Downloads/ieeg_password.bin' );
```

```
IEEGSETUP: Found log4j on Java classpath.
URL: https://www.ieeg.org/services
Client user: andrewc
Client password: ****
```

```
session.data


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

```
ans =

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

         snapName: 'I521_A0006_D001'
          montage: As Recorded
           filter: 'No Filter'
         resample: 'Not resampled'
       sampleRate: 2000
           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.
```

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

```
%set sample rate
sample_rate=2000;

%create filter
[b, a] = ellip(4,0.1,40, 300/(sample_rate/2),'high');
```

As computed by the code above, the denominator coefficients for my filter function are 1.000, -1.7432, 1.6167, -0.6559, 0.1430 the numerator coefficients for my filter function are 0.3420, -1.2740, 1.8676, -1.2740, 0.3420.

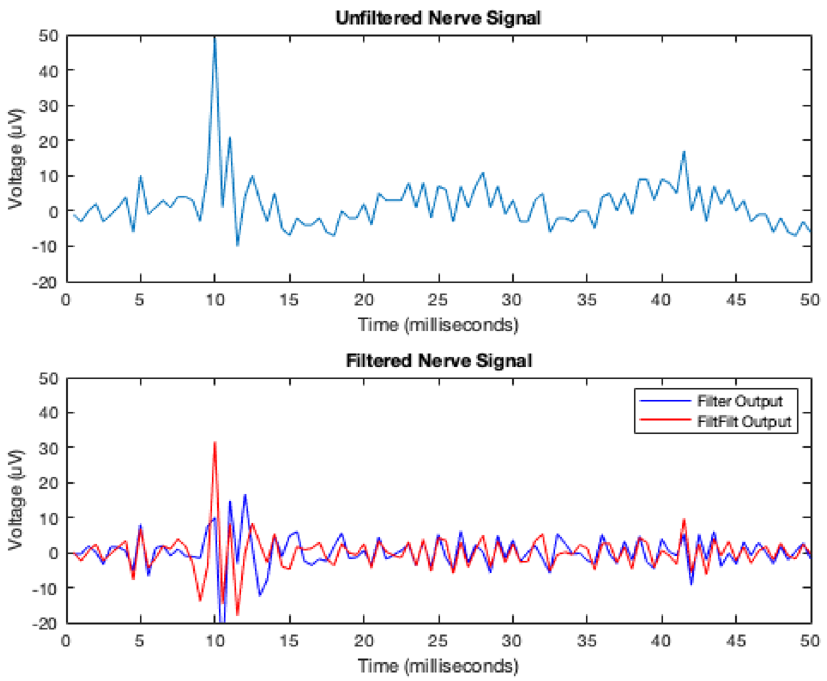
2. Using the **filter** function and **filtfilt** function, obtain two different filtered outputs of the nerve signal.

   (a) In a 2x1 subplot, plot the first 50 ms of the unfiltered nerve signal in the top subplot; in the bottom subplot, plot the **filter** output in blue and the **filtfilt** output in red. Use a potential range (y-axis) of -20 to 50 millivolts. (4 pts)

```
%create 50 ms time vector
time_ms=1/sample_rate:1/sample_rate:0.05;

time_ms=time_ms*1000;

%select 50 ms of nerve data
fifty_ms_nerve=(allData_nerve(1:100))/1000;
```

```
%do the filtering using the ellip function output

%filter function
filter_out=filter(b,a, fifty_ms_nerve);

%filtfilt function
filt_filt_out=filtfilt(b,a,fifty_ms_nerve);
```

```
%create subplot

figure
subplot(2,1,1)
plot(time_ms, fifty_ms_nerve)
ylim([-20, 50])
ylabel('Voltage (uV)')
xlabel('Time (milliseconds)')
title('Unfiltered Nerve Signal')
subplot(2,1,2)
plot(time_ms, filter_out,'b')
hold on
plot(time_ms, filt_filt_out,'r')
ylabel('Voltage (uV)')
xlabel('Time (milliseconds)')
title('Filtered Nerve Signal')
legend('Filter Output','FiltFilt Output')
ylim([-20, 50])
```



(b) How is the unfiltered signal different from the filtered signal? What is different about the two filtered (red and blue) signals? (2 pts)

The unfiltered signal is different from the filtered signal in that the points in the unfiltered signal's waveform have a higher amplitude (including much higher spikes) compared to the filtered signal. Additionally, the variation in the amplitudes of the points in the unfiltered wavefrom is higher than in the filtered waveform (in the filtered waveform the points are closer together to each other on the y-axis compared to the unfiltered waveform). The filtered (red and blue) signals are different in that the blue signal, generated using the MATLAB filter function, is phase shifted compared to both the red signal and the unfiltered data. However the red signal, generated with the MATLAB filtfilt function, is not phase shifted and the features of the signal in the red signal and the unfiltered data are at the same x-locations. Additionally, the red signal preserves the large positive voltage spike seen in the unfiltered data, while the blue signal cuts it down significantly. The amplitudes of the points in the red and blue signal are different throughout both waveforms (sometimes red is higher sometimes blue is higher). The blue waveform presents a significantly lower max voltage spike than the red waveform.

(c) Briefly explain the mathematical difference between the two filtering methods, and why one method might be more advantageous than the other in the context of spike detection? (5 pts)

The MATLAB filter function and filtfilt function differ in the way they filter input data. Specifically the MATLAB filter function represents a 1-D digital filter that uses a rational transfer function to filter the input data ("filter 1-D digital filter," 2020). However, the MATLAB filtfilt function does zero-phase digital filtering of the input data, and it filters the input data in the forward direction, reverses this now filtered data and puts it back through the filter ("filtfilt Zero phase digital filtering," 2020). Thus the filter created by the filtfilt function also has zero phase distortion, a transfer function equal to the squared magnitude of the original transfer function, and an order that is double the order specified by the transfer function numerator and denominator coefficients inputted ("filtfilt Zero phase digital filtering,"2020). The MATLAB filtfilt method might be more advatangeous compared to the MATLAB filter method in the context of spike detection because the filtfilt method does have zero phase distortion, which means the filtfilt method will preserve the features of the unfiltered data exactly where they occur in the filtered waveform ("filtfilt Zero phase digital filtering,"2020). However, the filter function phase distorts the input data in the filtered output. This is important for spike detection because the time at which a spike occurs may yield insight on the behavior it is decribing, the cause of the spike, and/or affect its detection. As such, it would be best to use the filtfilt method so the input data is not phase-distorted when it is filtered in the context of spike detection tasks.

References: filtfilt Zero phase digital filtering. (2020). Retrieved March 10, 2020, from https://www.mathworks.com/help/signal/ref/filtfilt.html
filter 1-D digital filter. (2020). Retrieved March 10, 2020, from

https://www.mathworks.com/help/matlab/ref/filter.html

3. Using a spike threshold of +30 mV, calculate the index and value of the peak voltage for each spike in the **filtered** nerve signal (select the best one). Use these values to plot the first 2.5 seconds of the nerve signal with a red dot above (e.g. 10 mV above) each spike. (Hint: Plot the entire length of the nerve signal with all the spikes marked but then restrict the x-axis using `xlim` to [0, 2.5] seconds) (4 pts)

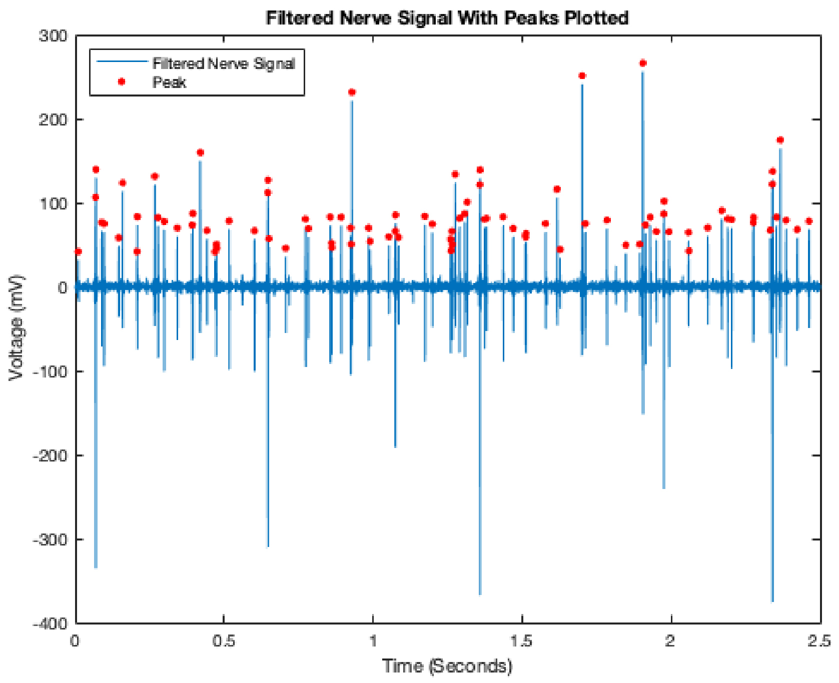
```
%run filt_filt on the whole dataset. filt_filt function is better than filter
%function

filt_filt_tot=filtfilt(b,a,(allData_nerve/1000));

%plotting 2.5 seconds of the filtered data

%create 2.5 second time vector
time_2=1/sample_rate:1/sample_rate:durationInSec+1/sample_rate;

%time_2=time_2*1000;

threshold = 30;

[pks, locs]=findpeaks(filt_filt_tot, time_2, 'MinPeakHeight', threshold)

%convert locatins to times in milliseconds
%loc_times=time_2(locs);
```

```
pks =

   31.6925
   96.3585
  129.4620
   66.4497
   64.9381
   48.1245
  113.5636
   31.6831
   73.4865
  121.3998
   72.2594
   67.6410
   59.7600
   63.3552
   76.9529
  149.7190
   56.7418
   31.5806
  40.2886
  36.1789
  68.2765
  56.5004
 101.9634
 116.8954
  47.0143
  35.8406
  70.4530
  59.1729
  72.8805
  41.7954
  36.3824
  72.7740
  60.0195
  40.3204
 221.5279
  59.9390
  43.9736
  49.2995
  56.1479
  75.1737
  48.7109
  73.7691
  64.5988
  46.4497
  32.8721
  55.9038
  40.0355
 123.7623
  71.2277
  76.5734
  90.5888
 111.2634
 129.0722
  70.4437
  71.1535
  73.1953
  59.3883
  51.2822
  49.0915
  53.4192
  65.0858
 105.9240
  34.1493
 241.0532
  64.9728
  65.1605
  69.1698
  39.4916
  40.5063
 255.9609
  63.4450
  72.8219
  55.5940
  76.5320
  91.6765
  55.4257
  54.5481
  32.4706
  60.1194
  80.5031
  70.6279
  69.8016
  66.2803
  72.4268
  57.2744
 111.8370
 127.2979
  72.7814
 164.6414
  68.9846
  58.0100
  67.9810
 134.9118
  69.4285
  39.5566
  57.7727
  61.2482
  51.0701
  55.7471
  58.4129
  53.1688
  78.6903
  68.3587
  31.7614
 103.4938
 138.1865
 169.3965
  61.8389
  50.2495
  69.5257
  58.9046
 233.0956
  63.9710
 217.2062
  55.4727
  46.7552
  56.2481
  60.0178
 238.9544
  35.7451
  36.9681
  66.3855
  68.3506
 121.6303
  59.4667
 110.7238
  68.8822
  61.1741
  61.7399
  42.4897
  34.7618
  74.2428
  60.3789
 110.7981
 115.7268
  59.4329
  42.4143
  73.8505
  71.2924
  74.4465
  40.3112
  64.7095
  31.0649
 241.0039
  54.5879
  74.4801
  72.7829
  60.9047
```

```
   56.9205
   99.6857
  115.6780
   31.7564
  229.7175
   49.4638
   43.8003


locs =

  Columns 1 through 7

    0.0100    0.0685    0.0695    0.0890    0.0970    0.1460    0.1590

  Columns 8 through 14

    0.2075    0.2090    0.2670    0.2775    0.2990    0.3425    0.3925

  Columns 15 through 21

    0.3945    0.4200    0.4420    0.4695    0.4735    0.4745    0.5165

  Columns 22 through 28

    0.6015    0.6465    0.6475    0.6500    0.7065    0.7725    0.7825

  Columns 29 through 35

    0.8560    0.8605    0.8615    0.8920    0.9245    0.9260    0.9290

  Columns 36 through 42

    0.9845    0.9895    1.0525    1.0745    1.0755    1.0850    1.1735

  Columns 43 through 49

    1.1990    1.2600    1.2610    1.2645    1.2655    1.2755    1.2900

  Columns 50 through 56

    1.3070    1.3155    1.3580    1.3590    1.3740    1.3805    1.4365

  Columns 57 through 63

    1.4700    1.5110    1.5120    1.5130    1.5785    1.6175    1.6275

  Columns 64 through 70

    1.7015    1.7110    1.7120    1.7850    1.8480    1.8945    1.9055

  Columns 71 through 77

    1.9140    1.9300    1.9505    1.9755    1.9765    1.9930    2.0585

  Columns 78 through 84

    2.0600    2.1230    2.1705    2.1900    2.2030    2.2755    2.2770

  Columns 85 through 91

    2.3330    2.3395    2.3405    2.3535    2.3665    2.3860    2.4235

  Columns 92 through 98
```

```
    2.4630    2.5245    2.5700    2.5980    2.6615    2.6845    2.7755

  Columns 99 through 105

    2.7770    2.8215    2.8855    2.9035    2.9715    2.9775    2.9820

  Columns 106 through 112

    2.9830    3.0235    3.0335    3.0575    3.0960    3.1595    3.1780

  Columns 113 through 119

    3.2100    3.2695    3.3130    3.3545    3.4080    3.4135    3.4470

  Columns 120 through 126

    3.4840    3.5160    3.5510    3.6050    3.6200    3.6210    3.6350

  Columns 127 through 133

    3.6705    3.7110    3.7270    3.7950    3.8020    3.8500    3.9035

  Columns 134 through 140

    3.9135    3.9145    3.9245    3.9630    3.9700    3.9965    4.0280

  Columns 141 through 147

    4.0870    4.0920    4.1185    4.1695    4.1735    4.2000    4.2380

  Columns 148 through 154

    4.2950    4.3005    4.3235    4.3245    4.3630    4.3745    4.4080

  Column 155

    4.4305
```

```
%create plot
figure
plot(time_2,filt_filt_tot)
hold on
for i=1:length(locs)
    plot(locs(i),pks(i)+10, 'r.','MarkerSize',10)
    hold on
end
xlim([0 2.5])
ylabel('Voltage (mV)')
xlabel('Time (Seconds)')
title('Filtered Nerve Signal With Peaks Plotted')
legend('Filtered Nerve Signal', 'Peak', 'Location', 'northwest')
```

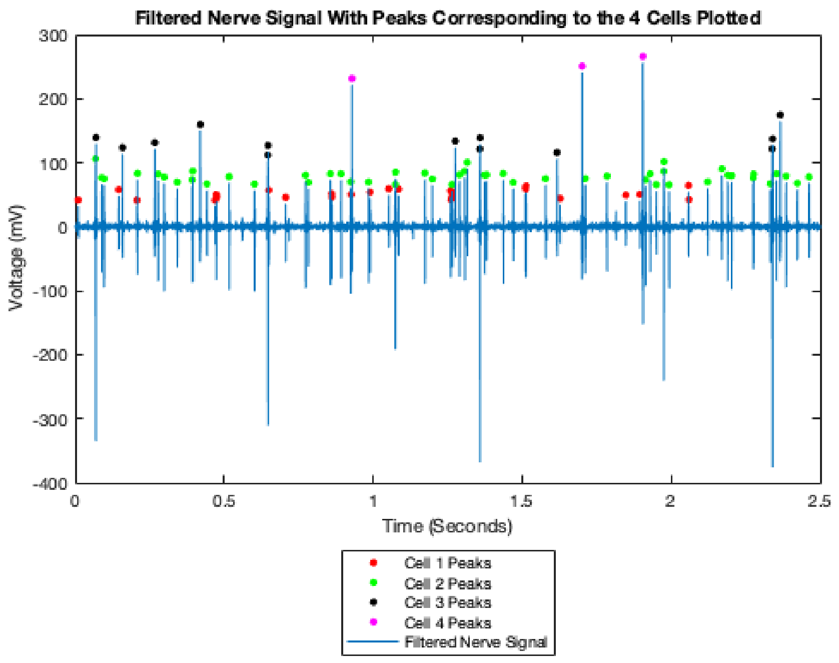


The filtfilt method was used to generated the plots above (the filtfilt method is the best method).

4. Under the assumption that different cells produce different action potentials with distinct peak amplitudes, decide how many cells you think were recorded (some number between 1 and 6). You may find it helpful to zoom in and pan on the plot you made in question 1.3. You may also find it useful to plot the sorted peak values to gain insight into where "plateaus" might be. (No need to include these preliminary plots in the report, though.) Use thresholds (which you well set manually/by eye) to separate the different spikes. Make a plot of the first 2.5 seconds similar to that in 1.3 except now color the spike dots of each group a different color (e.g., 'r.','g.','k.','m.').(6 pts)

```
%plot to visualize thresholds. no need to include it.
% figure
% yline(30)
% hold on
% yline(55)
% yline(100)
% yline(170)
% plot(time_2,filt_filt_tot)



threshold_1=30;
threshold_2=55;
threshold_3=100;
threshold_4=170;

[pks_1, locs_1]=findpeaks(filt_filt_tot, time_2, 'MinPeakHeight', threshold_1);
[pks_2, locs_2]=findpeaks(filt_filt_tot, time_2, 'MinPeakHeight', threshold_2);
[pks_3, locs_3]=findpeaks(filt_filt_tot, time_2, 'MinPeakHeight', threshold_3);
[pks_4, locs_4]=findpeaks(filt_filt_tot, time_2, 'MinPeakHeight', threshold_4);

figure
%Legend=cell(2,1);
```

```
for i=1:length(locs_1)
    h=plot(locs_1(i),pks_1(i)+10,'r.','MarkerSize',10);
    %Legend{1}=strcat('Cell 1 Peaks',num2str(1));
    hf(1)=h(1);
    hold on

end
for i=1:length(locs_2)
    k=plot(locs_2(i),pks_2(i)+10, 'g.','MarkerSize',10);
    %Legend{2}=strcat('Cell 2 Peaks',num2str(2));
    hf(2)=k(1);
    hold on
end
for i=1:length(locs_3)
    t=plot(locs_3(i),pks_3(i)+10, 'k.','MarkerSize',10);
    hf(3)=t(1);
    hold on
end
for i=1:length(locs_4)
    r=plot(locs_4(i),pks_4(i)+10, 'm.','MarkerSize',10);
    hf(4)=r(1);
    hold on
end

e=plot(time_2,filt_filt_tot);
hf(5)=e(1);
% %legend('Filtered Nerve Signal')
xlim([0 2.5])
ylabel('Voltage (mV)')
xlabel('Time (Seconds)')
title('Filtered Nerve Signal With Peaks Corresponding to the 4 Cells Plotted')
%legend(Legend)
%legend(p(1),'Cell 1 Peaks', h(1), 'Cell 2 Peaks')
legend(hf,{'Cell 1 Peaks','Cell 2 Peaks','Cell 3 Peaks','Cell 4 Peaks','Filtered Nerve Signal'}, 'Location',
```



5. Use Matlab's k-means[2] function (**kmeans**) to fit k clusters (where k is the number of cells you think the recording is picking up) to the 1D data for each spike.

   (a) Using the same color order (for increasing spike amplitude) as you did for the thresholds in question 1.4, plot the spike cluster colors as little dots slightly above those you made for question 1.4. The final figure should be a new plot of the nerve voltage and two dots above each spike, the first being your manual label and the second your clustered label, which (hopefully/usually) should be the same color. (4 pts)

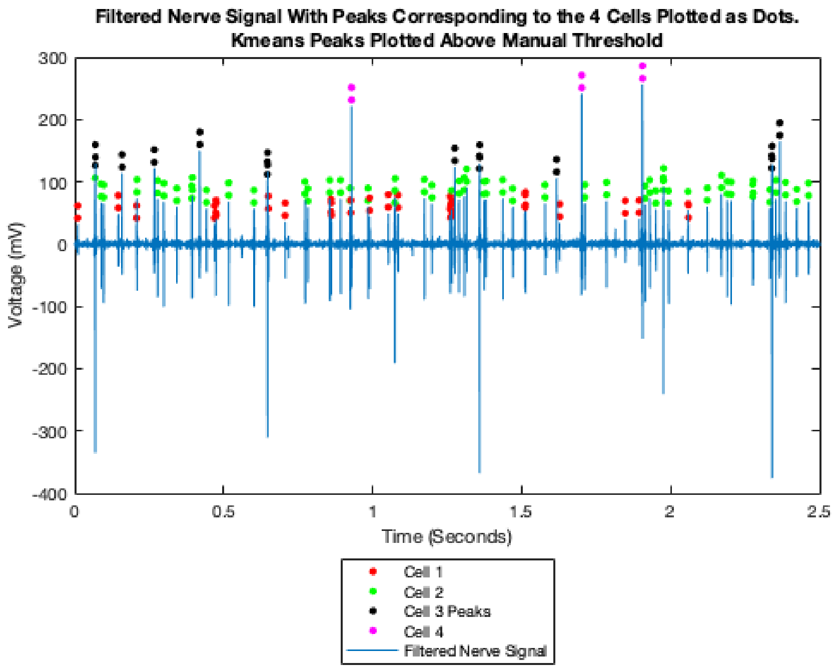
```
%concate all peaks into one vector
peaks_vector=[];

peaks_vector=vertcat(pks_1,pks_2, pks_3,pks_4);
```

```
%run matlab's k-means clustering
rng(1)% set the seed
idx=kmeans(peaks_vector,4);% this will group all the data into four groups.

all_locs=vertcat(locs_1',locs_2',locs_3',locs_4');

%create a mtrix with the flag, location, and peak for plotting
k_means_mat=[idx all_locs peaks_vector ];
```

```
%create plot
figure
%plot from 1.4
for i=1:length(locs_1)
    h=plot(locs_1(i),pks_1(i)+10,'r.','MarkerSize',10);
    %Legend{1}=strcat('Cell 1 Peaks',num2str(1));

    hf(1)=h(1);
    hold on

end
for i=1:length(locs_2)
    k=plot(locs_2(i),pks_2(i)+10, 'g.','MarkerSize',10);
    %Legend{2}=strcat('Cell 2 Peaks',num2str(2));
    hf(2)=k(1);
    hold on
end
for i=1:length(locs_3)
    t=plot(locs_3(i),pks_3(i)+10, 'k.','MarkerSize',10);
    hf(3)=t(1);
    hold on
end
for i=1:length(locs_4)
    r=plot(locs_4(i),pks_4(i)+10, 'm.','MarkerSize',10);
    hf(4)=r(1);
    hold on
end

%plot peaks from k means
for i=1:length(k_means_mat)
    if k_means_mat(i,1)==3
        plot(k_means_mat(i,2),k_means_mat(i,3)+30,'r.','MarkerSize',10);
        hold on
    elseif k_means_mat(i,1) ==4
```

[2]Clustering, like k-means you are using here, is a form of unsupervised learning.

```
        plot(k_means_mat(i,2),k_means_mat(i,3)+30, 'g.','MarkerSize',10);
        hold on
    elseif k_means_mat(i,1) == 1
        plot(k_means_mat(i,2),k_means_mat(i,3)+30, 'k.','MarkerSize',10);
        hold on
    elseif k_means_mat(i,1) ==2
        plot(k_means_mat(i,2),k_means_mat(i,3)+30, 'm.','MarkerSize',10);
        hold on
    end
end

e=plot(time_2,filt_filt_tot);
hf(5)=e(1);

xlim([0 2.5])
ylabel('Voltage (mV)')
xlabel('Time (Seconds)')
title({'Filtered Nerve Signal With Peaks Corresponding to the 4 Cells Plotted as Dots.', 'Kmeans Peaks 
legend(hf,{'Cell 1','Cell 2','Cell 3 Peaks','Cell 4','Filtered Nerve Signal'}, 'Location', 'southoutsid
```



(b) Which labeling, your manual ones or the ones learned by clustering) seem best, or do they both seem just as good? (Again, panning over the entire plot may be helpful.) (2 pts)

As can be seen from the dots in the plot generated in part 1.5a, the clustering labels are almost the same as the dots created by manual plotting most all of the points. We can see however there are some points where the two labels are different (for example at x = 2.05 seconds). While the clustering and manual labels are almost the same (and thus the performance of the two methods is almost the same), it appears that the manual labeling is better at doing the spike sorting (assigning the spikes to the correct cell) because the kmeans clustering unsupervised learning algorithim can at best provide an approximation of the sorting that the human eye can do. As such kmeans will not be as good as a human manually sorting the different spikes with thresholding. Additionally, manual sorting will not utilize any of the assumptions about the input data that are inherent in the k-means clustering algorithm, making it a more rigorous way to do labeling.

6. In this question, you will test the hypothesis that the muscle potential responses are really only due to spikes from a subset of the cells you have identified in the previous two questions. First, plot the first 2.5 seconds of the muscle fiber potential and compare it with that of the nerve. Observe the relationship between spikes and the muscle fiber response. (No need to include this plot and observation in your report.) Now, calculate the maximum muscle fiber potential change[3] in the 25 ms[4] window after each spike (with the assumption that spikes without any/much effect on the muscle fiber potential do not directly innervate it).

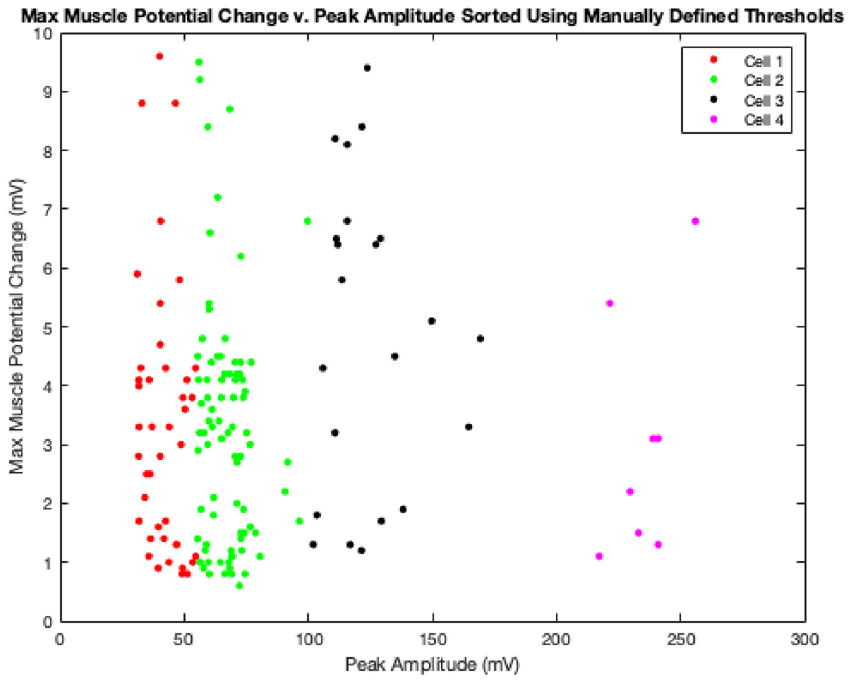
   (a) Using the cell groups you either manually defined or found via k-means clustering (just specify which you're using) again with different colors, plot a colored point for each spike where the x-value is the spike amplitude and the y-value is the muscle potential change. (6 pts)

```
%use the manual groups

%first plot 2.5 seconds of the muscle response and compare it with the
%nerve
% figure
% plot(time_2, allData_nerve/1000)
% xlim([0 2.5])
% figure
% plot(time_2, allData_muscle/1000)
% xlim([0 2.5])

mv_muscle_data=allData_muscle/1000;
%define all the peak times
all_peak_times=k_means_mat(:,2);

max_potential_change=[];%list of potential changes

%now calculate the maximum muscle fiber potential change (max voltage - min voltage) in the 25 ms
%window after each spike
for j=1:length(all_peak_times)
    %pull out 25ms = 0.025 seconds of muscle data. sample rate is 2000 Hz. so 0.025 sec of data
    %is 50 data points.
    index_1=find(time_2==all_peak_times(j));%define the first index
    voltage_window=mv_muscle_data(index_1:index_1+49);%define the voltage window
    max_potential_change(j)=max(voltage_window)-min(voltage_window);%calculate max potential change
end
%
% %create plot with the colors using the k means method
% figure
% for k=1:length(max_potential_change)
%     if k_means_mat(k,1)==3
%         plot(k_means_mat(k,3),max_potential_change(k),'r.','MarkerSize',10);
%         hold on
%     elseif k_means_mat(k,1) ==4
%         plot(k_means_mat(k,3),max_potential_change(k), 'g.','MarkerSize',10);
%         hold on
%     elseif k_means_mat(k,1) == 1
%         plot(k_means_mat(k,3),max_potential_change(k), 'k.','MarkerSize',10);
%         hold on
%     elseif k_means_mat(k,1) ==2
%         plot(k_means_mat(k,3),max_potential_change(k), 'm.','MarkerSize',10);
%         hold on
%     end
% end
% %plot(k_means_mat(:,3),max_potential_change,'.','MarkerSize',10)
% xlabel('Peak Amplitude (mV)')
```

[3] max voltage - min voltage

[4] Note that this 25 ms window is somewhat ad hoc and is just what seems reasonable by eye for this data. It implies no underlying physiological time scale or standard.

```
% ylabel('Max Muscle Potential Change (mV)')
% title('Max Muscle Potential Change v. Peak Amplitude Sorted Using K—Means Algorithm')

%create plot with the colors using the manual sorting thresholding method
figure
for i=1:length(locs_1)
    h=plot(pks_1(i),max_potential_change(i),'r.','MarkerSize',10);
    %Legend{1}=strcat('Cell 1 Peaks',num2str(1));
    gf(1)=h(1);
    hold on

end
for i=1:length(locs_2)
    k=plot(pks_2(i),max_potential_change(i+155), 'g.','MarkerSize',10);
    %Legend{2}=strcat('Cell 2 Peaks',num2str(2));
    gf(2)=k(1);
    hold on
end
for i=1:length(locs_3)
    t=plot(pks_3(i),max_potential_change(i+268), 'k.','MarkerSize',10);
    gf(3)=t(1);
    hold on
end
for i=1:length(locs_4)
    r=plot(pks_4(i),max_potential_change(i+298), 'm.','MarkerSize',10);
    gf(4)=r(1);
    hold on
end
%plot(k_means_mat(:,3),max_potential_change,'.','MarkerSize',10)
xlabel('Peak Amplitude (mV)')
ylabel('Max Muscle Potential Change (mV)')
title('Max Muscle Potential Change v. Peak Amplitude Sorted Using Manually Defined Thresholds')
legend(gf,{'Cell 1','Cell 2','Cell 3','Cell 4'})
```



(b) Does this plot support the hypothesis that the muscle fiber responses are only due to a subset of the cells. Explain why or why not. (3 pts)

No, this plot does not support the hypothesis that the muscle fiber responses are only due to a subset of cells. This is because we can clearly see that each cell (as grouped by manual sorting) creates a meaningful change in muscle potential compared to each of the other respective cells. Specifically cell 1 creates a change in muscle potential up to 9.6 mV, cell 2 creates a change in muscle potential up to 9.5 mV, cell 3 creates a change in muscle potential up to 9.4 mV, and cell 4 creates a change in muscle potential up to 6.8 mV. Thus, there were 4 cells recorded in this experiment and all of them contributed to a relatively large change in muscle potential and as such this plot does not support the hypothesis that the muscle fiber repsonses are only due to a subset of cells. The plot supports the hypothesis that the muscle fiber responses are due to all cells.

2 Multivariate Clustering (21 pts)

In this section, you will explore similar methods for spikes sorting and clustering but with a different dataset, the human intracranial data in `I521_A0006_D002`, which is a larger dataset of the same recording you saw in `I521_A0001_D001` of Homework 1.

1. Using a threshold six standard deviations above the mean of the signal, detect the spikes in the signal. In addition, extract the waveform from 1 ms before the peak to 1 ms after it with peak value in the middle. (You will end up with a matrix where each row corresponds to the number of data points in 2 ms of signal minus 1 data point. Use the closest integer number of data points for the ± 1 ms window.)

```
%create new session


session2 = IEEGSession('I521_A0006_D002', '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: ****
```

```
%session2.data;

sample_rate2=32258;
```

```
nr2 = ceil((session2.data.rawChannels(1).get_tsdetails.getEndTime)/1e6*session2.data.sampleRate);
```

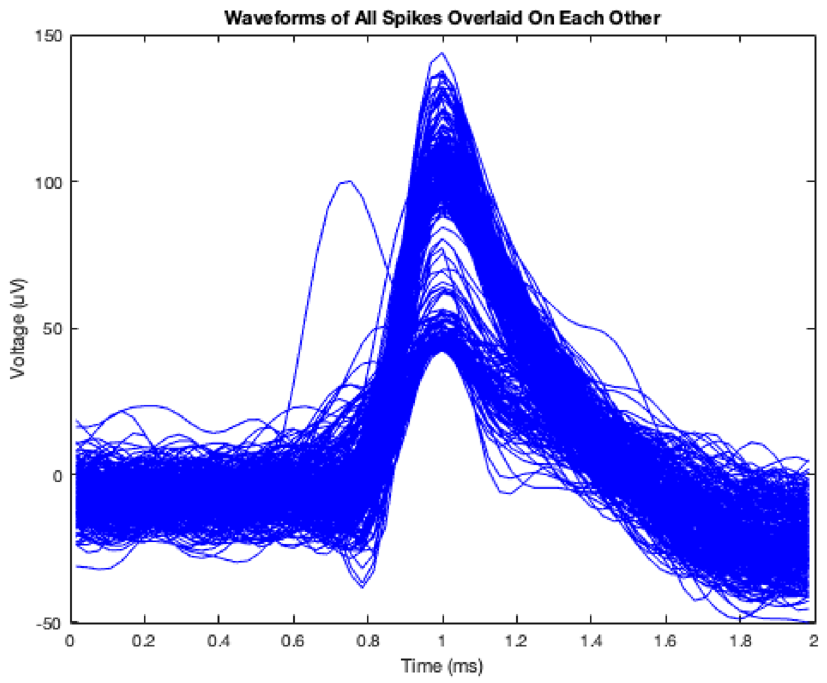
```
allData= session2.data.getvalues(1:nr2,1);
```

```
durationInUSec_2 = session2.data(1).rawChannels(1).get_tsdetails.getDuration;
durationInSec_2 = durationInUSec_2 / 1e6;%178.8449 seconds
```

```
%figure
%plot(allData)
%allData_mv=allData/1000;



mean_data=mean(allData);
std_data=std(allData);
threshold_p2 = mean_data + 6*std_data;

%run find peaks function to detect spikes in the signal
[pks_2, locs_2]=findpeaks(allData);
locs_2=locs_2(pks_2>threshold_p2);
pks_2=pks_2(pks_2>threshold_p2);


%[pks_p2, locs_p2]=findpeaks(allData, 'MinPeakHeight', threshold_p2);

%sampling rate is 32258 Hz, so 1 ms of data is 1e-3* 32258= 32.2580, rounds to 32
%datapoints of data

%extract the waveform from 1ms before the peak to 1ms after peak
wave_form_mat=zeros(length(locs_2),65);

for u=1:length(locs_2)
    index_left=locs_2(u)-32;
    index_right=locs_2(u)+32;
    wave_form_mat(u,:)=allData(index_left:index_right);
    %index_1=find(time_2==all_peak_times(j));%define the first index
    %voltage_window=mv_muscle_data(index_1:index_1+49);%define the voltage
    %window
end
```

(a) Plot the waveforms of all the spikes overlaid on each other in the same color. (4 pts)

```
%create 2.5 second time vector
%time_2=1/sample_rate:1/sample_rate:durationInSec+1/sample_rate;

time_plot=(1/65:(2/65):2);

%create plot
figure
for s=1:308
    plot(time_plot,wave_form_mat(s,:),'-b')
    hold on
end
xlabel('Time (ms)')
ylabel('Voltage (uV)')
title('Waveforms of All Spikes Overlaid On Each Other')
```



(b) Does it looks like there is more than one type of spike? (1 pt)

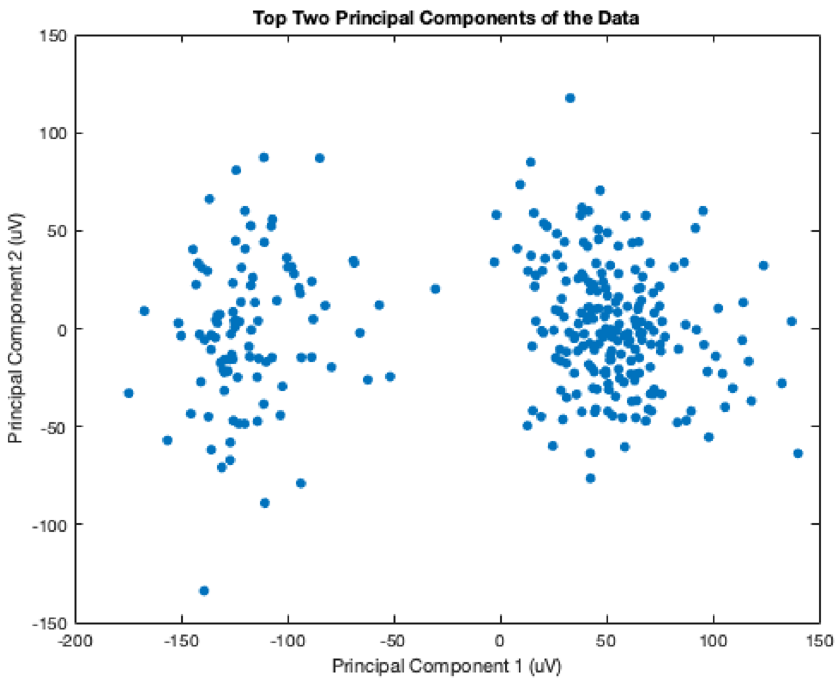
Yes it looks like there are two different types of spikes, one type looks like it has a higher amplitude than the other.

2. For each spike, represent the waveform by its principal components. Use the `pca` command in Matlab. Intuitively, principal component analysis finds the coordinate system that most reduces the variability in your data.

(a) Run principal component analysis on all the spike waveforms and represent your data with the top two principal components. Make a scatterplot of your data in this principal component (PC) space. (3 pts)

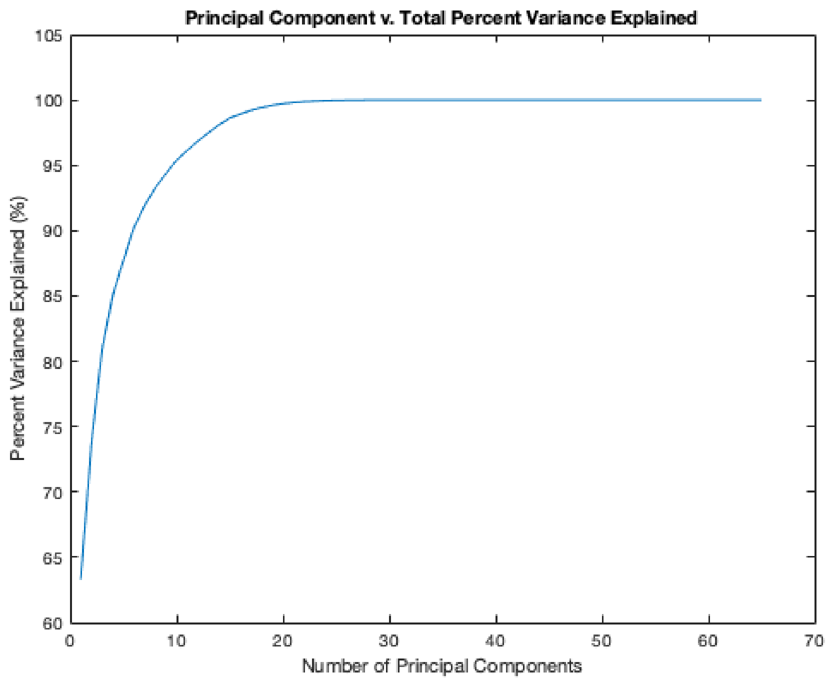
```
% Run PCA on all spike waveforms and represent data with top two principal
% components.

transposed_mat=wave_form_mat;

[coeff,score,latent,~,explained]=pca(transposed_mat);

%Represent data in the PC space
%PC_space=transposed_mat*coeff(:,1:2);
```

```
%create plot
figure
plot(score(:,1),score(:,2),'.','MarkerSize',14)
xlabel('Principal Component 1 (uV)')
ylabel('Principal Component 2 (uV)')
title('Top Two Principal Components of the Data')
% figure
% bar(latent)
```



(b) Each PC also has an associated eigenvalue, representing the amount of variance explained by that PC. This an output of the PCA command. Plot the principal component vs the total (cumulative) percent variance explained. What is the percent variance explained if you include the top two principal components? (3 pts)

```
explained_plotted=[];
for i=1:length(explained)
    value=explained(1:i);
    explained_plotted(i)=sum(value);
end

figure
%plot(explained,score,'b.','MarkerSize',10)
plot(explained_plotted)
xlabel('Number of Principal Components')
ylabel('Percent Variance Explained (%)')
top_two_explained=explained(1)+explained(2);
title('Principal Component v. Total Percent Variance Explained')
ylim([60 105])
```



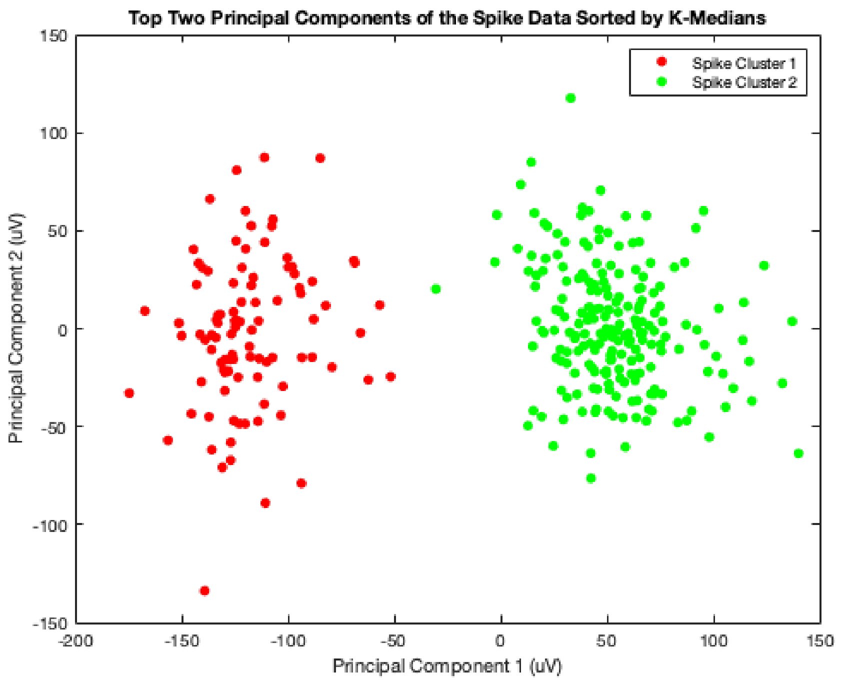
The top two principal components explain 73.7349 percent of the total variance.

(c) Does it look like there is more than one cluster of spikes? (1 pt)

Yes, it looks like there are more than one cluster of spikes. It looks like there are two clusters of spikes.

3. Use the same `kmeans` function as you used before to cluster the spikes based on these two (normalized) features (the waveforms represented by the top two PCs). You will use a slight twist, though, in that you will perform *k*-medians (which uses the medians instead of the mean for the cluster centers) by using the `'cityblock'` distance metric (instead of the default `'sqEuclidean'` distance). Make a plot similar to that in 2.2.a but now coloring the two clusters red and green. (3 pts)

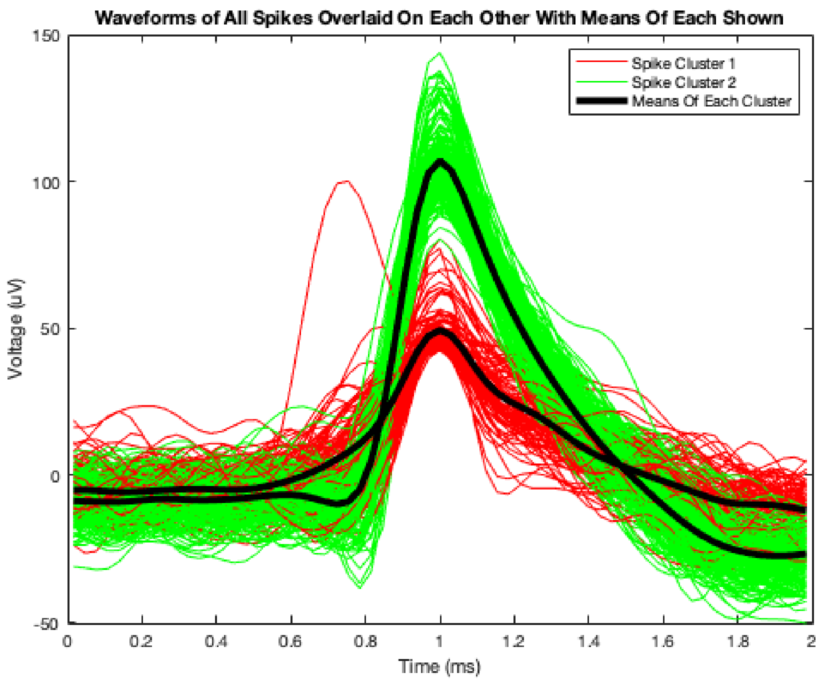
```
%run k-medians

idx_2=kmeans(score(:,1:2),2,'Distance','cityblock');

%create plot
figure
for i=1:length(score)
    if idx_2(i)==1
        q=plot(score(i,1),score(i,2),'r.','MarkerSize',14);
        qf(1)=q;
        hold on
    elseif idx_2(i)==2
        o=plot(score(i,1),score(i,2),'g.','MarkerSize',14);
        qf(2)=o;
        hold on
    end
end
xlabel('Principal Component 1 (uV)')
ylabel('Principal Component 2 (uV)')
title('Top Two Principal Components of the Spike Data Sorted by K-Medians')
legend(qf,{'Spike Cluster 1','Spike Cluster 2'})
```



4. Make a plot similar to 2.1 but now coloring the traces red and green according to which cluster they are in. Overlay the mean of the waveforms in each cluster with a thick black line (use the parameter `'LineWidth'` and value `'4'`). (3 pts)

```
%create plot with colored traces.

%figure out how many 1s and 2s there are in the idx_2
count_1s=0;
count_2s=0;
for i=1:length(idx_2)
    if idx_2(i)==1
        count_1s=count_1s+1;
    elseif idx_2(i)==2
        count_2s=count_2s+1;
    end

end
```

```
figure
cluster_1=zeros(count_1s,65);
cluster_2=zeros(count_2s,65);
for s=1:308
    if idx_2(s)==1
        e=plot(time_plot,wave_form_mat(s,:),'-r');
        cluster_1(s,:)=wave_form_mat(s,:);
        kl(1)=e;
        hold on
    elseif idx_2(s)==2
        z=plot(time_plot,wave_form_mat(s,:),'-g');
        %cluster_2=[cluster_2 wave_form_mat(s,:)];
        cluster_2(s,:)=wave_form_mat(s,:);
        kl(2)=z;
        hold on
    end
end

cluster_1_filtered=cluster_1(any(cluster_1,2),:);
cluster_2_filtered=cluster_2(any(cluster_2,2),:);
mean_cluster_1=mean(cluster_1_filtered,1);
hold on
mean_cluster_2=mean(cluster_2_filtered,1);
hold on
q=plot(time_plot,mean_cluster_1,'k-','LineWidth',4);
kl(3)=q;
hold on
plot(time_plot,mean_cluster_2,'k-','LineWidth',4)
xlabel('Time (ms)')
ylabel('Voltage (uV)')
title('Waveforms of All Spikes Overlaid On Each Other With Means Of Each Shown')
legend(kl,{'Spike Cluster 1','Spike Cluster 2','Means Of Each Cluster'})
```



5. What are some dangers of using the clustering techniques in this homework? (List 3) (3 pts)

   Some of the dangers of using the clustering techniques in this homework include: First, k-means clustering assumes that each cluster present in the input data should have roughly the same about of datapoints in it (Robinson, 2015). This is a danger because if the clusters actually have different numbers of datapoints the k-means clustering unsupervised learning algorithm may group the data non-optimally, leading to inaccurate conclusions. Second, the behavior of the k-means clustering algorithm is also greatly effected by the presence of outliers in the input data, when outliers are present the output clusters may be created non-optimally leading to a poor representation of the data (Franklin, 2019). A third danger that the input data to the k-means clustering technique used in this homework should actually have some inherehent clusters in it or else k-means clustering will not applicable; that is if k-means clustering is run on a uniform data (non-clustered) dataset the k-means clustering algorithm will partition the data into clusters, even though the clusters would be meaningless in this case (Kim, 2015).

References: Franklin, J. (2019, November 16). Effect of outliers on K-Means algorithm using Python. Retrieved March 10, 2020, from https://medium.com/analytics-vidhya/effect-of-outliers-on-k-means-algorithm-using-python-7ba85821ea23

Kim, K. (2015, February). How to understand the drawbacks of K-means. Retrieved March 10, 2020, from https://stats.stackexchange.com/questions/133656/how-to-understand-the-drawbacks-of-k-means

Robinson, D. (2015, January 16). K-means clustering is not a free lunch . Retrieved March 10, 2020, from http://varianceexplained.org/r/kmeans-free-lunch/