EE 486E: Homework 2

Spring 2015

50 points

Due: MON 02 FEB

Directions for submission

Turn in a paper copy of your writeup at the beginning of class on the due date. Your writeup should consist of typed, numbered answers to each question followed by an addendum containing only the **PLOTS** requested in all-caps bold red font, followed by your code. Include two plots per page, each with a title indicating which problem it answers and axes labels that have proper units. In addition, upload your *Matlab code only* to Google Drive (`EE486E_CLASS\EE486E_DROPBOX\<YOURNAME>_EE486E_DROPBOX`). Your code may be split into more than one m-file but it must have a single main file, `RunHw<X>_<YOURNAME>.m`, that calls all of the others.

1 Features and Simulations

An intial "feature extraction" step, in which raw data are summarized by a series of computed metrics, forms the backbone of almost all detection strategies, from seizures in EEG to faces in images. Features are often defined as equations; the task for someone who wishes to employ a given feature in an algorithm is to implement it 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.

1.1 Consider the line-length feature, which simply computes the sum of the magnitudes of the differences between successive elements of a sequence of n points, $\mathbf{x} = (x_1, x_2, \ldots, x_n)$,

$$LL(\mathbf{x}) = \sum_{i=2}^{n} |x_i - x_{i-1}|$$

This feature actually has a very simple implementation in Matlab: `sum(abs(diff(x)))`. Consider the following toy signal: a 100-dimensional signal (i.e., 100-element vector) comprised of a single square pulse of amplitude 2 that begins at time index 50 and ends at time index 80 (where time index 1 is the first time point). Values before point 50 and after point 80 are zero.

(a) Without using Matlab, what is the line length of this toy signal? (1 pt)

(b) Construct the square pulse described above. Then **PLOT** the *cumulative* line length at each time point (i.e., the line length of the sub-signal $\mathbf{x}^{(m)} = (x_1, \ldots, x_m)$, for $m = 2, \ldots, n$; see Matlab's `cumsum` function). Before you make this plot, though, consider what it *should* look like and then see if your plot confirms your intuition. (This practice not only ensures

that you're thinking a bit more about what you're doing but also makes detecting bugs in your code easier.) (2 pts)

1.2 Implement each of the following features for the same square pulse signal. For each feature, **PLOT** the cumulative feature value at each time point (three separate plots). You should be able to implement both the Rectified Area feature and the Energy feature in one statement, whereas the Entropy feature may take a few more statements. (3 pts each, 9 pts total)

(a) Rectified Area, $RA(\mathbf{x}) = \sum_{i=1}^{n} |x_i|$

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

(c) Entropy,

$$H(\mathbf{x}) = -\sum_{j=1}^{b} P(y_j) \log P(y_j)$$

where $P(y_j)$ is the probability of a signal sample falling into a discrete amplitude bin with center value y_j. For our purposes, we'll use a 5-bin histogram (which means $b = 5$; see Matlab's `hist` function) and find the total number of values that fall into each bin. The quotient of the counts in each bin j and the total number of samples gives us an estimate of the probability $P(y_j)$, a number between 0 and 1, of a value falling into bin j. Use the natural logarithm and assign any occurences of log(0) a value of 0 (see the `isinf` function for handling this).

1.3 You will now test these four features on two other simulated signals. For each, use a sampling rate of 100 Hz and a duration of 10 seconds. (Note: your first time point should be 0 and last should be 9.99.)

(a) Construct an amplitude-modulated (AM) signal by modulating a 1 Hz sine-wave with the triangular pulse $2\text{tri}(\frac{1}{5}(t-5))$. This can be accomplished using Matlab's `triang` function. (How? Element-wise multiply your sinusoidal signal by `2*triang(length(x))'`, where `x` is your sinusoidal signal and `'` is the transpose operator.) **PLOT** the AM signal. (2 pts)

(b) Show the cumulative feature **PLOT** of for each of the four features (4 plots). Again, think about what you'd expect for each feature before you actually do the plot. (4 pts total)

(c) Construct a frequency-modulated sine-wave signal (a linear chirp) with amplitude 2 by plotting $2\sin(2\pi f t^2)$ with $f = 0.1$ Plot the signal. (2 pts)

(d) As before, show the cumulative feature **PLOT** for each of the four features. (4 pts total)

2 Feature Overlays

In the last section, you plotted the accumulated values of various features over time to get a sense for the sensitivities of each feature to different signal characteristics. While such cumuluative feature plots are sometimes used in their own right in BMI applications, it is perhaps more typical to compute the feature value at a given time point using a time-window of fixed duration. The window is then shifted along the time-axis by some pre-specified duration in order to generate the

feature-value at the next time point, and so forth. In this section, you will take this sliding window approach using the line-length feature, and overlay the resultant feature time-series on a segment of EEG containing a seizure. This data is stored in `szChirps.mat` with variables `info` and `data`.

2.1 Under the right-aligned convention, what number of time-series samples would be ignored from the left in terms of the window length L and window displacement d and record length n? (2 pts)

2.2 For this data, how many samples from the left will be ignored with a 4-second window and 1/8 second displacement? (2 pts)

2.3 You will use the provided functions `rfeature.m` and `zohinterp.m` to produce a time-aligned, line-length feature vector for the original EEG data, using a 4-second window length and 1/8 second window displacement. The `rfeature.m` function takes a function name or function handle as its second argument. (Search Matlab's help documentation for "anonymous functions" if you're curious to learn more about the code below and function handles). For data vector `data`, you can use the commands

```
[llFeat, numIgnored] = rfeature(data,@(x) sum(abs(diff(x))),L,D);
temp = [NaN(1,numIgnored+L-1) zohinterp(llFeat,D)];
llFeatInterp = temp(1:length(data));
```

which will a) calculate a line-length feature vector using window length `L` and displacement `D` and then b) interpolate the vector (using a zero-order hold) to make it the same length as the data vector. Normalize the line-length feature values to have a maximum twice that of the original EEG signal's maximum. Then **PLOT** the entire signal in blue and overlay the aligned line-length feature in green. Include a figure legend. (4 pts)

2.4 What threshold might you use on the *raw* line-length feature vector (not the one scaled to the EEG signal) in order to capture the 17 most prominent pre-seizure "chirps" that occur? (1 pt)

2.5 Using this threshold value, find the leading points in time where the threshold is crossed. Make a new **PLOT** identical to the one from question 2.3, only now including red vertical lines at these threshold crossing times. Give the lines a length that spans the range of the EEG data. These red event-markers should indicate the pre-seizure chirps, the seizure onset, and some flickering during the end of the seizure. (2 pts)

3 Predictions with Features of Features

Often "features of features" are used in seizure detection and prediction applications. In this section, we will explore generating one such "meta-feature" as the ratio of the mean and variance of another feature. You will use new EEG data in the file `multiSz.mat` with variables `info`, `data1`, and `data2`. Plot the signal in `data1` and note the times when you think the two seizures begin.

3.1 Use `rfeature` to compute the value of the energy feature you implemented in question 1.2, using a sliding window with width 10 seconds and displacement 1 second. Interpolate the result as above using `zohinterp` and **PLOT** it overlayed on the original signal. As in question 2.3, normalize the feature time-series to have a maximum of twice the EEG signal maximum. (2 pts)

3.2 Now use the syntax `rfeature(energyFeat,@mean,L,D)` to compute a meta-feature: the sliding average (30-second window width, 2-second displacement) of the feature signal you generated in 3.1 (where `energyFeat` is the latter signal), and interpolate it appropriately. Use an analagous approach, with the same window parameters, to generate an interpolated sliding average variance signal. Finally, compute the mean-to-variance ratio at each time point, and overlay this ratio signal, after normalizing as above, on a **PLOT** of the seizure. (4 pts)

3.3 Does this ratio feature look like good candidate for a seizure detector? If so, what threshold might you set (on the non-normalized feature vector) for it? (2 pts)

3.4 The signal in `data2` contains another seizure (whose location should again be fairly obvious). **PLOT** the data along with the ratio feature (normalized for the signal in `data2`) and the threshold you determined in the previous question. You can represent the threshold with a horizontal black line; just be sure to apply the `data2` normalization to the threshold before plotting. (3 pts)

3.5 Does the ratio feature look similar to that for the EEG of `data1`? Did you detect the seizure? Did you detect any other extraneous events? In a few sentences, discuss how well you think this ratio feature coupled with a fixed threshold might perform on other seizures. (4 pts)