ECSE 517
Prof. Sam Musallam
Fall 2016
Due Nov 3 2016

Sort action potential waveforms using principal component analysis (PCA). To help visualize the PCA process, we will apply it to data that has 2 dimensions. But keep in mind, PCA and similar techniques are very important to process the high dimensional data.

Imagine a dataset uniformly and *tightly* clustered around the line y=2*x. Thus, each point is described by an x-coordinate and a y-coordinate. Using your magical abilities, you decide to rotate the dataset clockwise around the origin until the data becomes clustered uniformly and *tightly* around the x-axis. After the rotation, the y-coordinate can be discarded eliminating a dimension. Note the importance of the word "tightly", which ensures that the incurred error of discarding a dimension was low. If the dataset was *loosly* distributed around the line y=2*x, then discarding a dimension may yield unacceptable errors. This is an oversimplification but thinking about these concepts in low dimensions may help you understand them. Let's generate some random data to visualize the example above:

The unbiased variance of a 1-D sample dataset called *x* is defined by:

$$var(x,x) = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^T(x_i - \bar{x})$$

(if you had data for all the population, then the 1/(n-1) becomes 1/n.) The *T* means "transpose", which in this case is meaningless since xi-xbar is a scalar (but we'll need it later). Similarly, if you had two 1-D sample datasets called *x* and *y* , then the covariance between *x* and *y* is:

$$cov(x,y) = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^T(y_i - \bar{y})$$

As you can see, the variance of x is just the covariance of x with itself. The covariance tells you how the variables change relative to each other (when x increases, does y increase too? And so on).

Define 2 random datasets S1 and S2. S1 will be uncorrelated while S2 will be correlated. Matlab commands can be in red or italics. In Matlab, type:

S1=randn(1000,2); define 1000 X 2 points of normally distributed data with mean=0 and variance =1. We will let the first column be the x-axis and the 2nd column be the y-axis. This is the uncorrelated set.
For S2:

```
S2(1:1000,1)=S1(:,1);
```

```
S2(1:1000,2)=S1(:,1)+ randn(1000,1);
```

The second column of S2 is a function of the first column. This is the correlated set.
Plot the two datasets. `figure; subplot(2,1,1); scatter(S1(:,1),S1(:,2),4);`

```
subplot(2,1,2);
scatter(S2(:,1),S2(:,2),4);
```

Calculate the covariance of each set:

```
cv1=cov(S1); cv2=cov(S2);
```

cv1and cv2 are 2X2 matrices. The diagonal of the covariance matrix is the variance of each dimension since diagonal elements are the covariance of the respective dimension with itself. The off-diagonal elements are the covariances. Note that the covariance matrix is symmetric (covariance(x,y)=covariance(y,x)) Since S1 was generated randomly, the covariance between the two columns should be close to zero. Therefore, uncorrelated data has zero covariance. However, since S2 has correlated dimensions, then the off-diagonal elements should be nonzero[1].

Let's relate this to PCA by finding the principal components manually (without using commands like "pca" or "pcacov"). PCA attempts to find a dimension that lies along the direction of maximum variance. From the plot of S2 below, we can see that the direction of maximum variance is a line going through the data diagonally (see red line in figure 1 below).

[1] the covariance(x,y)/sqrt(var(x))*sqrt(var(y)) is the correlation coefficient between x and y

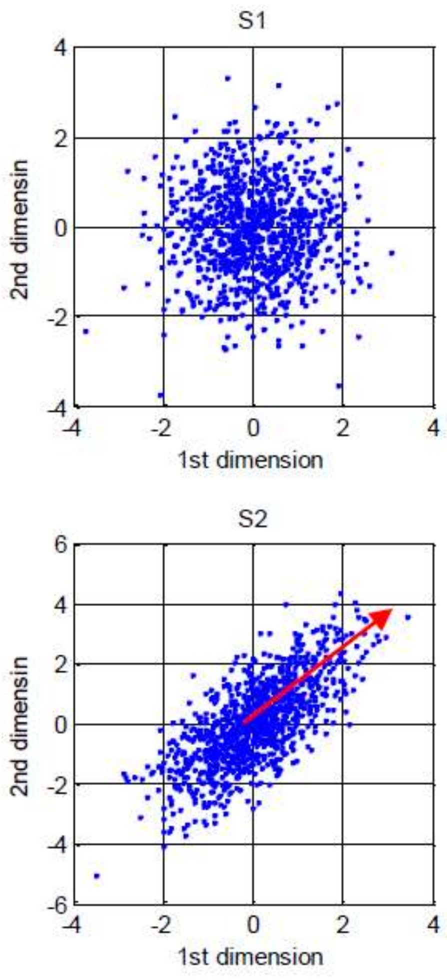

Figure 1. Plot of S1 (top plot) and S2 (bottom plot). Note that your plots will look different than mine since the data is randomly generated. The direction of maximum variation in the bottom plot is along the red line, diagonally through the data. This is because this direction has the largest range.

The red line in Figure 1 is an *eigenvector of the covariance*. The corresponding *eigenvalue* is an indication of how much variance is explained. The largest eigenvalue is called the 1st principal component (PC), the 2nd largest is the 2nd PC and so forth. To analytically find the red line drawn in figure 1 (bottom plot), type:

[EVec,Eval]=eig(cv2) % your output will be different than mine since the data is random.

```
EVec =

  -0.8492    0.5281
   0.5281    0.8492


Eval =

   0.3785         0
0    2.6461
```

The eigenvalues are returned in Eval. Since 2.6461 is the largest eigenvalue, then it is the 1st principal component. It is important to sort the eigenvalues in descending order to facilitate the analysis. The corresponding eigenvector is returned in the rows of EVec. The eigenvector corresponding to the largest eigenvalue for my data originates at (0,0) and points towards (.5281,.8492) – right along the direction we guessed (the plot is too small to check this but please verify that the line described by EVec is indeed the line we seek) . Plot this vector on your S2 plot to convince yourself that it points in the right direction. Note that the eigenvector is normalized. To extend the line to cover all the data, just multiply EVec by some constant:

The amount of variance explained by each principal component is the ratio of that PC to the sum of all PCs. So the first PC captures (2.6462/(2.6461+.3785)) *100 %, or 87% of the variation and so on. This means that 87% of the variation occurs along the first eigenvector (dimension with largest range). Two or three dimensions often account for over 90% of the variance in most multidimensional datasets. As the number of dimensions increases, it helps to plot the variance explained (manually or with a Pareto plot). If you let PCnorm=PC./sum(PC), then pareto(PCnorm) returns a curve of cumulative explained variance superimposed on a histogram showing the variance explained by each PC. Or you can just plot the cumsum(PC)/sum(PC).

Part A: Dimensionality Reduction

Question 1: Spike Sorting. Please submit all code and all requested plots. The data for both assignments is called Data1 and contains spikes recorded on a single electrode from dorsal premotor cortex during a memory reach trial. Each waveform is 1ms long. Other fields are described as needed.

1. Clean up data for spike sorting.
   a) Load Data1 into Matlab. Two identical waveforms offset in voltage or shifted in time will not have the same PC. Therefore, remove any offset in the data and align the waveforms by shifting them back or forward in time (no more than 3 time increments). First, plot a sample of waveforms (if you have a slow computer, do not plot all of them).
      ```
      Waves=Data1.Waves;
      Plot(Waves(1:100:end,:)’); grid on % note the transpose.
      ```

The plot command above will plot every 100th waveform. Note the waveforms have similar shapes. To align the all the waves, 1) find the mode of the minimum of all waves, 2) Shift all waves backward or forward in time (maximum of 3 steps) so that all the waveforms have the same minimum. Below, I refer to the aligned waveforms as *Awaves* See command *circshift*

2. Run pca on Awaves.
   a) Plot the cumulative variance accounted by each PC (variance explained by the current PC and PC's before it. See above discussion). You can calculate the explained variance or use a pareto on one of the variables returned by *pca* (see above),
   b) When is 90% of the variance accounted for? How about 95%? How many PCs account for less than 0.1% of the variance?

3. Use a scatterplot to plot PC1 vs PC2. Adjust the size of the data points to make the cluster separation apparent. Note that the distance between points across different clusters can be less than the distance between points contained in a single cluster. (This is bad for clustering algorithms.) How many clusters can you count? For datasets with small cluster separation, it may help to view the PCs as a surface plot. See code below. (You can submit one or both plots).

```
edg=prctile(scores(:,1:2),[0.1,99.9]);
De=min(diff(edg))/30;  e1{1}=[edg(1,1):De:edg(2,1)];
e1{2}=[edg(1,2):De:edg(2,2)];
  hst=hist3(scores(:,1:2),e1);
Shand=pcolor(e1{1},e1{2},hst');
set(Shand,'LineStyle','none');
```

4. Use *kmeans* to cluster the data. Pass *kmeans* the number of clusters you counted in the previous question. Separate the waveforms into independent groups according to the membership indices returned by *kmeans*. If you counted N clusters, you should generate N units. Also group the variable Data1.Tw using the group indices. Tw contains the waveform time (spike time). Plot each unit separately. Comment on the quality of the separation.

Part B: Decodes using spikes

1. Data1 also contains fields called T0 and T19. These vectors contain the start and end of each trial. For each unit generated above, create a matrix of spike times by finding the spike times between T0(i) and T19(i) for trial i. Also create a matrix of spike times for the original unsorted data. Below is pseudocode that will help you. Name each matrix spkTrialsi where i=1 to number of units.

```
% initialize spkTrialsI
  For all sorted units including the original unsorted data, do:
    for i=1:length(T0);
        spkInTrial= find spikes in unitITs between T0(i) and T19(i)
        tsByTrial=unitITs1(spkInTrial);
        if ~isempty(tsByTrial)
        spkTrialsI(i,1:length(spkInTrial))=tsByTrial-tsByTrial(1);

        end
    end
```

2. Plot the tuning curve for all grouped units and for the original data set. Reach direction is in Data1.Hang. Data1.Tmem1 contains the time = 100ms+Memperiod start. Data1.Tmem2 contains the time = 800ms + Memperiod start. For each trial k, find the firing rate (in spikes/sec) between Tmem1(k) and Tmem2(k). For each unit i, label the firing rate vector FRMemi. Use the *errorbar* command to plot the mean number of spikes in the memory period sorted by direction. Use the standard error of the mean (SE) for each direction as the error vector in *errorbars*. The SE is the standard deviation of the mean and equals the std(fr)/sqrt(number of trials). Are the preferred directions of all grouped units the same? What is the meaning of the tuning curves? Test the null hypothesis that each unit is tuned to reach direction. Use anova1(FRmemI,Hang,'off'). (Bonus Question: why can't you use a ttest to test the null hypothesis?)

3. Plot a *post-stimulus time histogram (PSTH)* for all the units using trials when Hang=45 degrees. One way to compute the PSTH is to convolve the spike train with a kernel. For the kernel use a normal distribution with a standard deviation of 20ms and one with a standard deviation of 70ms. Use *normpdf* to generate the distribution. The easiest way to generate the PSTH is to transform the spike times in spkTrialsI into matrices of 1's and 0's (where 1 represents an action potential) and then convolve each trial with the kernel. How does the standard deviation effect the plot in the time domain? How does it affect the frequency content of the signal.

4. Use the FRMemI vectors to decode the reach direction. decode reach direction using each sorted unit separately, and then decode using combination of sorted units, including a decode using all sorted units together. To use more than one unit, you have to build a matrix of firng rates where each column is a unit. Use the classify function to run the decode. This function can be used with different discrimination functions. Read the matlab help on the discrimination functions and choose the one that optimizes the success rate (you can try them all and just choose the best one). Report the decode success rate in a table. Repeat the decode using the unsorted units. Did spike sorting improve the decode? Why or why not? Report the mean success rate for each run.

See below for pseudocode

```
Folds=4;
ind = crossvalind('Kfold',ones(length(Hang),1),Folds);
for different combinations of sorted units  for j =
1:Folds     test = (ind == j);     train = ~test;
    Xtest=test trials from fr.
    Xtrain=train trials from fr
    Ytrain= trials form Hang to train classifer.
    Ytest=test trials from Hang to decode Xtest
class = classify(Xtest,Xtrain,Ytrain,type );
    classPerf(j)=length(find((class-Ytest)==0))./length(class);
    end mean
of Folds std
of Folds.
Repeat for different number of units.
```