Lab 7
CO902 – Probabilistic and statistical inference – 2012-13 Term 2
Lecturer: Tom Nichols
Clustering with PCA

A basic technique in text mining is a "bag of words" approach. In the spirit of naïve Bayes, the order in which words appear in a document is ignored, and we only consider the frequency at which words occur. In `NYtimes.mat` is data on the frequency of 4425 words in 102 articles that appeared in the New York Times. This particular selection consists of 57 stories about art and 45 stories about music[1]. Perform unsupervised clustering with PCA, using the "art" vs. "music" labels to see if the clustering is informative.

The NYtimes.mat contains 5 variables

`X`	: 4425 × 102 matrix of word frequencies
`Words`	: length-4425 cell array of words
`WordsSp`	: length-4425 cell array of words, each appended by a space
`Article`	: length-102 cell array of article types, 'music', or 'art'
`ArtLab`	: length-102 string array, 'm' or 'a', useful for plotting

1. Performa a PCA analysis on the word frequencies X using a SVD decomposition to avoid creating a 4425 × 4425 covariance matrix. Plot the eigenvalue spectrum and the cumulative eigenvalues. How many non-zero eigenvalues are there? (Do you know why!?) How many components are needed to accurately reconstruct this data? How many components would you examine for interesting structure?

2. Examine the words that have the largest (most positive) weights on the first PC, and those that have the smallest (most negative). Is the PCA finding any structure in the data? What about on the 2nd, 3rd, etc, PC's. Can you 'make a story' about what each PC is capturing? (Remember that the Matlab function `sort` can return a vector of indices reflecting the sort order; you can use this to easily find the elements of `WordsSp` that correspond to extremes of the PC. Also, use horizontal concatenation of the `WordsSp` cell array to view a subset of the words conveniently, e.g. to see the first 10 words: `[WordsSp{1:10}]`

3. Compute the reduced data using the Principal Components (i.e. use U to project `X` into Y). Make scatter plots of the first few components vs each other, and then apply plotting symbols based on the articles. One way to do this is with the Matlab function `text` to plot text on an existing plot; a useful trick is to create an "empty" plot where the plotting symbols are white, and then use the text command to plot at those locations. E.g.
   ```
   plot(1:10,1:10,'w');text(1:10,1:10,num2str([1:10]'))
   ```
   Do the PC's capture the difference between the two types of articles? Which PC's?

4. *Time permitting.* Build a classifier to predict article type. What sort of accuracy can you achieve? What level of data reduction dimensionality (if any) is optimal?

[1] Data prepared by Cosma Shalizi of CMU Statistics, http://www.stat.cmu.edu/~cshalizi/uADA/12. Note that the values in `X` are adjusted as per an "inverse document-frequency weighting" scheme, where highly frequent words (e.g. "a", "the") are down-weighted to emphasize more meaningful words.