Data Mining: 36-462/36-662
Homework 1

Due Tuesday February 5 2013
(at the beginning of lecture)

Append your R code to the end of your homework. In your solutions, you should just present your R output (e.g., numbers, table, figures) or snippets of R code as you deem it appropriate. Make sure to present your results (i.e., your R ouput) in a clear and readable fashion. Careless or confusing presentations will be penalized.

Problem 1

Download the file "hw1prob1.Rdata" from the course website, and load it into your R session using `load("hw1prob1.Rdata")`. Now you'll have a document-term matrix `dtm` loaded into memory, of dimension 8 documents $\times$ 6820 words. These are the 8 wikipedia documents that were used in lecture 2—4 of them on the teenage mutant ninja turtles and 4 of them on the artists of the same name. If you're curious, type `rownames(dtm)` and `colnames(dtm)` (don't worry if the second command gives you a warning).

(a) First normalize the documents by their total word count, and then apply IDF weighting to the words. Save this matrix as `dtm1`. Now reverse the order: first apply IDF weighting to the words, then normalize the documents by their total word count, and save this matrix as `dtm2`. Are `dtm1` and `dtm2` different? Explain why you think they should or shouldn't be different.

(Note: your normalization and IDF weighting must be computed manually, i.e., you can't use functions in the `tm` package.)

(b) We're going to forgo IDF weighting with such a small collection (8 documents). Normalize the documents by their total word count, and call this matrix `dtm3`. According to this document-term matrix, which document is closest (measured in Euclidean distance) to the document named "tmnt mike"?

(c) Sticking with the normalized matrix `dtm3`, compute the distance between each pair of documents using the function `dist`. Now run hierarchical agglomerative clustering, both with single linkage and complete linkage, using the function `hclust`. Plot the resulting dendograms for both linkages. If you had to split the documents into 2 groups, which linkage do you think gives a more reasonable clustering?

(d) Combine the word counts from all of the documents into one cumulative word count vector. I.e., for each word, you should now have a count for the number of times it appears

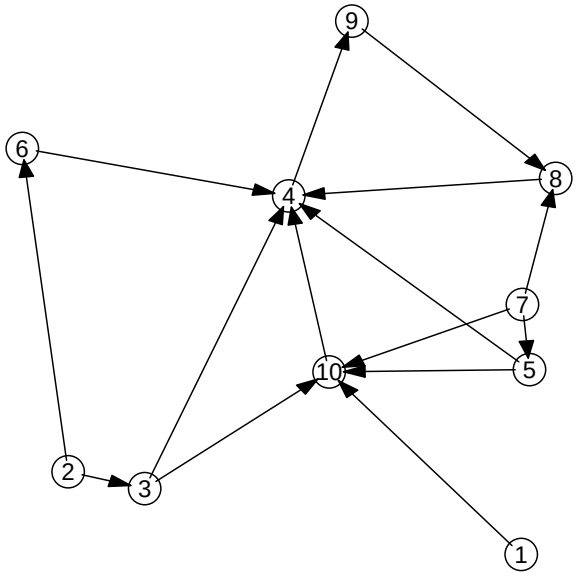
across all 8 documents. List the top 20 most common words, and how many times they appear. What percentage of total occurrences do these top 20 words represent? How many of the top words account for 50% of total occurrences?

(e) Zipf's law states that, given a collection of documents like the one we have, the number times a word appears is inversely proportional to its rank (the words being ranked by how common they are). In other words, the second most common word appears half as often as the most common word, the third most common word appears a third as often as the most common word, etc. This "law" is one that has been empirically confirmed, and aside from word counts, these kind of "power" laws have also been observed in a wide variety of different problems.[1] Does our collection of 8 wikipedia articles appear to follow Zipf's law? Can you give a plot to provide visual evidence for or against this claim?

(Hint: for your plot, think about translating the relationship expressed by Zipf's law into a mathematical one, between the number of occurrences y and the rank x of a word. Now take logs.)

Problem 2

Compute the PageRank vector for the following graph, with $d = 0.85$. Repeat the calculation for $d = 1$ (BrokenRank). What's the difference? Explain.



You should be computing PageRank by explicitly constructing A and finding its leading eigenvector (i.e., you can't use the `page.rank` function in the `igraph` package).

(Hint: you don't have to compute the leading eigenvector by repeatedly multiplying by A, although you can if you really want to. Instead, use the `eigen` function in R.)

[1] Professor Shalizi would have something to say about this: `http://arxiv.org/pdf/0706.1062v2.pdf`.

Problem 3

You're going to prove some claims from lecture 4.

(a) Let $X_1, \ldots X_n \in \mathbb{R}^p$, and C be a function assigning points to clusters $1, \ldots K$. Let n_k be the number of points assigned to the kth cluster. Prove that the within-cluster scatter here is exactly the within-cluster variation:

$$\frac{1}{2}\sum_{k=1}^{K}\frac{1}{n_k}\sum_{C(i)=k}\sum_{C(j)=k}\|X_i - X_j\|_2^2 = \sum_{k=1}^{K}\sum_{C(i)=k}\|X_i - \bar{X}_k\|_2^2,$$

where $\bar{X}_k$ is the average of the points in group k, i.e., $\bar{X}_k = \frac{1}{n_k}\sum_{C(i)=k} X_i$ (here I mean vector addition).

(Hint 1: don't let the vectors and norms scare you! Prove the result for scalars first, and then see what you can do.)

(Hint 2: don't let the multiple sums above scare you! Get rid of the outer sum taken over $k = 1, \ldots K$, and consider a fixed group k.)

(b) Let $Z_1, \ldots Z_m \in \mathbb{R}^p$. Prove that the quantity

$$\sum_{i=1}^{m}\|Z_j - c\|_2^2$$

is minimized by taking $c = \bar{Z}$, the average of the points, i.e., $\bar{Z} = \frac{1}{m}\sum_{i=1}^{m} Z_i$ (again this is vector addition).

(c) Let W_t denote the within-cluster variation at the start of iteration t of K-means clustering. Prove that for any t, $W_{t+1} \leq W_t$.

Problem 4

In this problem you're going to investigate the invariance of agglomerative clustering using either single or complete linkage under a monotone transformation of the distances.

Download the file "hw1prob3.Rdata" from the course website and load it into your R session using `load("hw1prob3.Rdata")`. Now you'll have two objects loaded into memory:

- `x`, a 40×2 matrix containing 40 observations along its rows;
- `d`, the pairwise Euclidean distances.

(a) Run hierarchical agglomerative clustering with single linkage, using the function `hclust`. Cut the tree at $K = 4$ clusters using the function `cutree`, which returns a vector of cluster assignments. Plot the points in `x` with different colors (or different `pch` values) indicating the cluster assignments. Also plot the dendogram.

(Note: if you want your dendograms to look like the ones in lecture, choose a really small negative value for the `hang` parameter, e.g., `hang=-1e-10`).

(b) Repeat part (a) for complete linkage.

(c) Repeat parts (a) and (b), but passing `d^2` to the function `hclust` instead of `d`. Did the clustering assignments change? Did the dendograms change?

(d) Prove that for single linkage, running agglomerative clustering with dissimilarities d_{ij} and running it again with dissimilarities $h(d_{ij})$ produces the same sequence of clustering assignments, provided that h is a monotone increasing function. Recall that a monotone increasing function h is one such that $x \leq x'$ implies $h(x) \leq h(x')$. Prove the same thing for complete linkage.

(e) Run agglomerative clustering with average linkage on each of `d` and `d^2`. Cut both trees at $K = 4$. Are the clustering assignments the same? How about for $K = 3$? (You should produce plots of `x`, like the ones you made above, with different colors or `pch` values in order to display the clustering results.)

Bonus problem

Write your own K-means function in R. It should have the following form:

```
my.kmeans = function(x, centers, max.iter=20) { ... }
```

where it takes the arguments:

- `x`. an $n \times p$ matrix with the n observations along its rows;
- `centers`, a $K \times p$ matrix giving the K initial guesses for the cluster centers;
- `max.iter`, a number (defaults to 20) giving the maximum number of iterations before quitting.

It should return:

- `centers`, a $K \times p$ matrix containing the K final cluster centers;
- `cluster`, a vector of length n indicating the cluster assignment for each point;
- `num.iter`, the number of iterations taken by the algorithm.

As a check, compare your algorithm `my.kmeans` to the standard `kmeans` function in R, by passing both functions the same data points and the same initial centers. (Remember to use `algorithm="Lloyd"` in `kmeans`.)