Problem Set 7

Assigned: Apr. 17
Due: Apr. 24

## Problem 1

Consider the following collection of data points in two dimensions:

| | A | B | C | D | E | F | G | H | I | J | K | L |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| x | 1 | 10 | 1 | 5 | 4.5 | 9.1 | 2 | 5 | 7 | 2.5 | 8.5 | 8 |
| y | 10 | 10 | 9.1 | 0 | 1 | 8.9 | 8.5 | 2 | 6 | 7 | 8.5 | 9 |

Trace the behavior of the k-means algorithms, with $k = 3$, starting from the centers $u = \langle 5, 10\rangle$, $v = \langle 4, 7\rangle$ and $w = \langle 7, 7\rangle$. Your trace should show the alternation of computing center points and assignments of points to clusters.

At each stage of the algorithm — that is, each time the cluster assignments are computed and each time the new center points are computed — compute the value of the cost function:

$$Cost(C) = \sum_{p \in S} D^2(p, C(p))$$

In the above formula:
$S$ is the set of all the points.
$C$ is the mapping from data points to the associated center point.
$D$ is the Euclidean distance, and $D^2$ is the square of the Euclidean distance.
Thus $D^2(\langle p_x, p_y\rangle, \langle q_x, q_y\rangle) = (p_x - q_x)^2 + (p_y - q_y)^2$

Do not simply do a web search for code for computing k-means and execute it. However, short of that, you can use computational tools to whatever degree you want with this, including writing your own program to compute k-means (Seems obvious, but I've occasionally had students who for some reason thought I wanted them to do all the computations by hand.) My own preference is to use an interpreted languages (Python or Matlab) in desk-calculator mode to compute matrices with all the distances squared, compute the centers, and add up the cost function, and to get the minimum values and construct the clusters manually by eyeballing it; but use whatever style you want.

## Problem 2

Apply the agglomerative clustering algorithm to points A−G from problem 1. Take the distance between two clusters A and B to be the maximal distance between any point in A and any point in B.

$$d(A, B) = \max_{x \in A, x \in B} d(x, y)$$

You should show the tree that the algorithm generates. Label the interior nodes of the tree with the order in which the algorithm creates the node. Break ties arbitrarily.

Again, you may use whatever computational tools you want, short of searching on the web for an algorithm that solves the entire problem.