1 More Clustering

1.1 Finding Cliques from Last Time

Recall from last time, we wanted to find a solution to the following problem. Given a graph G, the adjacencies of two vertices v_1, v_2 in G is toggled with some probability $p < \frac{1}{2}$ to give us a new graph G'. Our only knowledge is the structure of G'. The problem is to reconstruct with "some certainty" where the cliques (maximal connected subgraphs) of the original graph G were. Ideally, one would look for principal submatrices (submatrices which are closed under reflection about the diagonal) of the incidence matrix for G' which are "dense in 1s" and "sparse in 0s". If the vertices happen to be ordered so the cliques have vertex sets which are contiguous with respect to this ordering, then this problem is "fairly easy": we need only look at square blocks around the diagonal, and see when these are "dense in 1s" and "sparse in 0s". If not (which is more likely the case), we would need to examine all principal submatrices, which is equivalent to looking at the square blocks around the diagonal for each possible permutation of the vertex ordering. Since there are lots of permutations, this isn't feasible in practice.

1.2 Other Clustering Heuristics

Recall from last time, we discussed one non-heirarchical clustering method, the "core-based" method. In this class, we discuss two additional (unstructured) clustering heuristics, K-means and **S**elf-**O**rganizing **M**aps. We also start to discuss Model Based Clustering.

1.2.1 K-Means

The basic idea for K-means is to begin with our data points segregated into K clusters, and through each pass "migrate" points to more appropriate clusters.

- **K-means Clustering**
- **Input:** A collection of points in d-dimensional space, and $K > 0$ (K is the number of clusters desired).
- **Output:** A set of $\leq K$ clusters.

1. Initialize. Can be done in a variety of ways. Can be done naively by randomly distrubuting the points among K sets.
2. For each cluster C, calculate the centroid for C, meaning, calculate the average of each of the d coordinates among the points in C.
3. With the centroids calculated from (2), for each point, decide which centroid it is closest to, and reassign the point to the cluster corresponding to this centroid.
4. Repeat until some iteration count is reached, or until some sort of convergence is reached.

Some comments about the K-means heuristic:

- After the first pass, the data points are fairly well aggregated about the different centroids.
- The resulting clusters may be heavily dependent on the initialization step. Namely, if a different collection of random points is chosen at the beginning, we may end up with a different clustering partition at the end.
- In effect, this heuristic strives to minimize the total intra-cluster distance.
  Meaning, minimize:

$$\sum_{clusters\ C} \Big(\sum_{(i,j) \in C^2} |x_i - x_j|^2 \Big) = \alpha \sum_{clusters\ C} \Big(\sum_{i \in C} |y_i|^2 \Big)$$

  (α a constant) where $|y_i| = |x_i - \bar{x}^{(C(i))}|$ is the distance from x_i to the centroid of the cluster it is in. ($\bar{x}^{(C(i))}$ is the centroid of the cluster containing data point i). In essence, this is intuitively clear: we'd want to be able to move the centroids as close as possible to the data points.
- This solution is not heirarchical. It is more of a global optimization problem...minimizing the sum above over all possible collections of K clusters. Since there are scads of collections of K clusters, (and minimizing sums of this type may be **NP**-hard?), we can attempt to get as close as our patience allows us by setting the iteration limit.
- The clusters we get depends on K, specifically, as K increases, the diameter of the clusters decreases. So, when K passes the number of data points, we can put each data point in its own cluster, so the above sum is 0.
- It is possible that we end up with less than K clusters by the time termination occurs. Perhaps, during one pass, for each data point in a specific cluster, there is a different centroid which that point is closer to. Then, after this pass, the cluster having these points originally will no longer have any points in it.

1.2.2 Self-Organizing Maps

The other clustering method described is a bit more technical. It doesn't exactly seem as much of a "clustering" problem as it seems like a "find the best K centroids" problem. Comments were made that in light of this, seemingly "similar" data were forced into different categories for K large enough. The basic setup for the self-organizing map heuristic is pretty flexible though.

- **Self-Organizing Maps**
- **Input:** Functions α, σ, integer $K > 0$, a collection of data points, and some underlying topology (with K points).
- **Output:** K centroids. Specifically, a centroid corresponding to each point in the topology.

A couple of comments/requirements on the setup:

- $$\lim_{t\to\infty} \alpha(t) = \lim_{t\to\infty} \sigma(t) = 0$$
- $\alpha(t)$ represents (roughly) "how much a centroid is allowed to be moved" on pass t.
- $\sigma(t)$ represents (roughly) "how much we want to move centroids connected in the topology" on pass t.
- In lecture, the underlying topology used for illustrative purposes was a grid.

With all of that in mind, the following describes the **SOM** process:

1. Set $t = 1$. Initialize. Again, many ways to do this. Could be done naively by picking K random data points as the centroids and putting them at "appropriate" places in the topological structure (where, more arguments could be made as to good was to do this step).
2. For each data point x (currently in cluster C_x with centroid $\bar{x}$), find the closest centroid to x, and assign x to the corresponding cluster C'_x. Update the centroid (old_center to new_center) for each cluster C as follows:

   If x was in C originally, and $C'_x = C$, leave centroid alone.

   Otherwise, new_center = old_center + $h(t)(x$ - old_center), where

   $$h(t) = \alpha(t)e^{-d^2/(\sigma(t))^2}$$

   d = the topological distance from cluster C'_x to cluster C. (on a 2D grid, this was the taxi-cab distance between the two points on the grid).
3. Increment t. If t "large enough", stop. Otherwise goto (2).

There are many variants on this heuristic. There is a lot of flexibility in the initialization stage, in the underlying topology we use, the number of clusters we desire, the functions α and σ. Comments were made that the specific choice of α and σ didn't seem crucial. Suggestions were made that perhaps instead of restricting the topology from the outset of the algorithm, perhaps instead build a directed graph based on the centroids chosen in the initialization stage, and then impose whatever topological constraints you want on this digraph instead (planarity, k-regularity, etc.) The argument for this idea included: algorithms for these classes of digraphs have been more thouroughly researched, and by imposing the topology before we actually have the data, we may be prematurely forcing conglomeration or segmentation of clusters.

1.2.3 Model Based Clustering

The last type of clustering discussed (and will be gone into more in depth in the future) was model based clustering. Where all of the previous clustering methods discussed made no stringent assumptions on the input data, model based clustering will make some sort of assumptions. Specifically, the input data may be assumed to have some well known statistical/mathematical distribution, be it normal, binomial, Poisson, or whatever. Further, it may be known ahead of time that the data arose from K different stoichastic (random) processes, with some hypothesized distribution. Then under these assumption, the goal is to determine with some probabilistic certainty the mean, median, variance, and whatever other statistical parameters are desired for each of the K processes. Then, with these parameters "determined", given some data point, we wish to determine the probability it arose from each of the K processes. This will give us a form of *Soft Clustering*, where data points are not partitioned into sets, but instead associated with different processes with some probability. This is in contrast to the *Hard Clustering* performed by all of the methods described thus far: the data points are partitioned into some collection of sets. The method that will be used (and discussed) is the **EM**-algorithm, which will perform much the same as K-means under the assumption that the K processes have spherical symmetry and constant variance.