Clustering

Last time …

- PageRank
  - Topic-sensitive
  - Link Spam

- Today
  - Clustering

Points, Spaces & Distances

- Points
- Spaces (normed vector space)
  - e.g. Euclidean
  - $\|x\| > 0, \text{ if } \ x \neq 0$
  - $\|\alpha x\| = |\alpha|\|x\|$ for any scalar α
  - Triangle inequality, $\|x + y\| \leq \|x\| + \|y\|$
- Distance measure
  - non-negative
  - Symmetric
  - Triangle inequality

Clustering

- Hierarchical or agglomerative algorithms
- Point assignment

- Euclidean or arbitrary distance measure (*e.g.* Riemann, Hyperbolic)
  - Centroid of cluster
- Will the data fit in main memory

- Dimensionality

Hierarchical Clustering

- Start with n clusters, with 1 point each
- While (not stopping criteria)
  - pick two clusters to merge
  - Merge clusters and update centroid

- Each iteration is $\mathcal{O}(n^2)$
  - $\mathcal{O}(n^3)$
  - $\mathcal{O}(n^2 \log n)$

- Use fast distance computations → BSP / octrees → $\mathcal{O}(n \log n)$

Non-Euclidian Spaces

- Need to pick appropriate distance measure
- Centroid not appropriate for clustering
  - Choose a sample in place of centroid
- Hyperplane partitions

K-means clustering

- Point assignment algorithm
- Assume Euclidean space
- Assume number of clusters, k, is known in advance

1. Choose k points that are likely to be in different clusters
2. Foreach remaining point p do
    1. Find the cluster (centroid) closest to p
    2. Add p to this cluster
    3. Update the centroid for the cluster

Initializing the clusters

- Pick points that are as far away from one another as possible
  - Pick a random first point
  - Add additional points that maximize the minimal distance to already selected points
- Cluster a sample of the data and choose clusters
  - Random samples
  - Hierarchical clustering

Bradley, Fayyad & Reina (BFR)

- High-dimensional Euclidean space
- Clusters are normally distributed about the centroid (assumption)

- Start by choosing k centroids
- Read data in chunks
- Keep 3 types of information in memory
  - Discard Set → $2d + 1$ values (number of points, sum, sum of squares)
  - Compressed Set → $2d + 1$ values
  - Retained Set

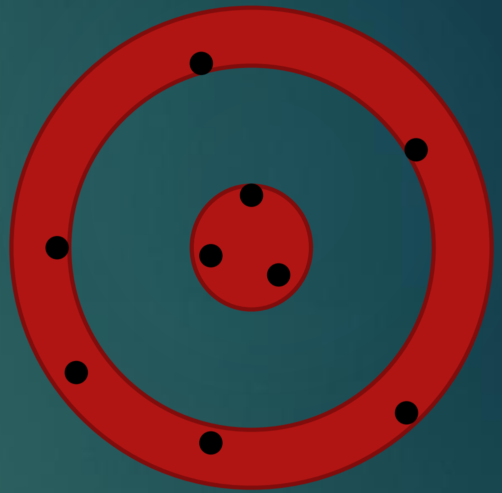
Bradley, Fayyad & Reina (BFR)

- First, all points that are sufficiently close to the centroid of a cluster are added to that cluster (Mahalanobis distance)
- For the points that are not sufficiently close to any centroid, we cluster them, along with the points in the retained set
- Merge miniclusters (new clusters + existing compressed set)

- Final processing of points in the retained set and the miniclusters in the compressed set
  - Discard → outliers
  - merge

Clustering Using REpresentatives

- CURE
- Assumes Euclidean space
- No assumptions on the shape of clusters
- Uses a collection of representative points



1. Sample data and cluster hierarchically
2. Select representative points for each cluster
3. Move each of the representative points a fixed fraction of the distance between its location and the centroid of its cluster
4. Merge clusters if close

Clustering in non-Euclidean spaces

Clustering in non-Euclidean spaces

- Use a combination of hierarchical and point-assignment
- Represent clusters by sample points in memory
- Organize clusters hierarchically in a tree
- Insert new points by traversing the tree

Representing Clusters (Ganti et al.)

- Clusters represented by a set of features
  - N, total number of points in the cluster
  - Clusteroid → point in the cluster that has minimum cumulative squared distance to all other points
  - The cumulative squared distance (CSD) of the clusteroid
  - k closest points to the clusteroid and their CSD
    - New clusteroid will be from one of these
  - k furthest points to the clusteroid and their CSD
    - Needed to decide whether to merge two clusters

Initializing the tree

- Tree structure is similar to a B-tree
- Each leaf node holds as many clusters as possible
- Interior nodes holds a sample of the clusteroids on its subtrees

- Initialize by hierarchically clustering a sample of the data
  - Choose only clusters that are of a specific size
    - These are the leaf nodes
  - Group clusters based on common ancestors
- Balance tree if necessary

Adding points

- Read points from disk and traverse down tree, using distance to clusteroid (samples)
- On reaching the leaf, pick and update the cluster representation
    - Increment N
    - Update the CSD of all $2k + 1$ points
        - Squared distance to the new point
    - Estimate the CSD of the new point, p
        - $\text{CSD}(p) = \text{CSD}(c) + Nd^2(p, c)$
    - Check if p is either the k nearest or farthest point
    - Check if one of the k nearest points is the new clusteroid

Clustering in a Streaming model

- Assume a sliding window of N points
- We are interested in the clusteroids of the best clusters formed from the most recent m points, for any $m \leq N$
- Pre-cluster the points so that queries can be answered
-

Stream-Clustering (Babcock et al.)

- Partition and summarize data into buckets
  - Bucket sizes are powers of two → $c\ 2^k$
  - Only one or two of any size
  - Bucket sizes are restrained to be non-decreasing as we go back in time
    - $\log N$ buckets
- Contents of a bucket
  - Size of the bucket
  - Timestamp of the bucket → most recent point
  - Cluster representations (multiple clusters per bucket)
    - Number of points in the cluster
    - Clusteroid
    - Other info needed to merge and maintain the clusters

Initializing buckets

- p - smallest bucket size
- Every p points, create a new bucket. Timestamp the bucket
  - Cluster points

- Drop any bucket older than N
- If we have 3 buckets of size p → merge the two oldest
- Might have to propagate merges ($2p, 4p$ ...)

Merging buckets

- Size of new bucket is twice as large
- Timestamp of the new bucket is the newer of the two being merged
- Decide on whether to merge clusters

- Consider: k-means, Euclidean
  - Represent clusters using number of points (n) and centroid (c)
  - Choose $p = k$, or larger → k-means clustering while creating bucket
  - To merge, $n = n_1 + n_2$, $c = \frac{n_1 c_1 + n_2 c_2}{n_1 + n_2}$

Merging buckets

- Size of new bucket is twice as large
- Timestamp of the new bucket is the newer of the two being merged
- Decide on whether to merge clusters

- Consider: non-Euclidean (Ganti et al.)
  - Represent clusters using clusteroid and CSD
  - Need to choose new clusteroid while merging
    - Will be one of the k-points furthest from the clusteroids
    - $CSD_m(p) = CSD_1(p) + N_2\left(d^2(p, c_1) + d^2(c_1, c_2)\right) + CSD_2(c_2)$

Answering queries

- Given m, choose the smallest set of buckets that covers the most recent m points. At most $2m$ points
- Merge clusters