Hierarchical Clustering

Why hierarchical clustering?

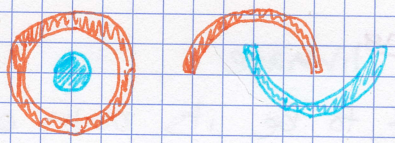
- Avoid choosing # clusters beforehand
- Dendrograms help visualize different clustering granularities
  - No need to rerun algorithm
- Most algorithms allow user to choose any distance metric
  - k-means restricted us to Euclidean distance

Can often find more complex shapes than k-means or Gaussian mixture models

k-means: spherical clusters

Gaussian mixtures ellipsoids

What about these?



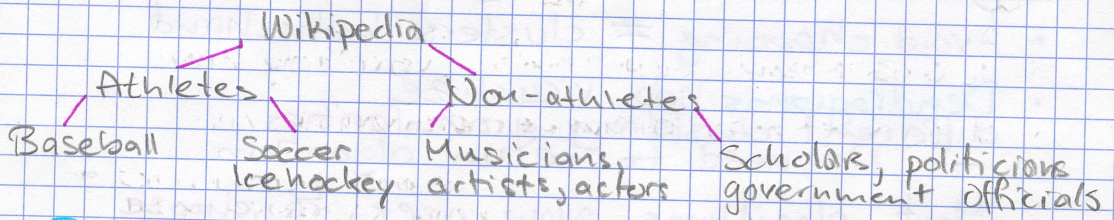
Two main types of algorithms:

Divisive, aka top-down: Start with all data in one big cluster and recursively split
- Example: recursive k-means

Agglomerative aka bottom-up: Start with each data point as its own cluster. Merge clusters until all points are in one big cluster
- Example: single linkage

Divisive clustering

Divisive: Recursive k-means



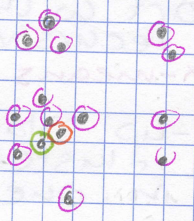
Divisive choices to be made

- Which algorithm to recurse
- How many clusters per split
- When to split vs. stop
  - Max cluster size: number of points in cluster falls below threshold
  - Max cluster radius: distance to furthest point falls below threshold
  - Specified # clusters: split until pre-specified # clusters is reached

Agglomerative clustering

Agglomerative: Single linkage

1. Initialize each point to be its own cluster
2. Define distance between clusters to be:



$$distance(C_1, C_2) = \min_{x_i \text{ in } C_1,\ x_j \text{ in } C_2} d(x_i, x_j)$$

d: specified pairwise distance function

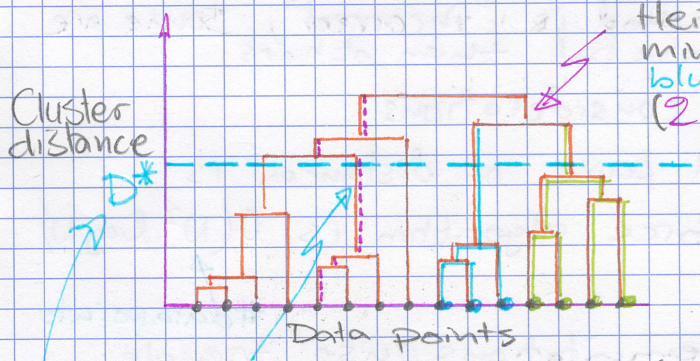
Linkage criteria

3. Merge the two closest clusters
4. Repeat step 3 until all points are in one cluster

The dendrogram

The dendrogram for agglomerative clustering

- x axis shows data points (carefully ordered)
- y axis shows distance between pair of clusters



Path shows all clusters to which a point belongs and the order in which clusters merge

Extracting a partition

Choose a distance D^* at which to cut dendrogram

Every branch that crosses D^* becomes a separate cluster

Agglomerative clustering details

Agglomerative choices to be made

- Distance metric: $d(x_i, x_j)$
- Linkage function: eg, $\min_{x_i \text{ in } C_1,\ x_j \text{ in } C_2} d(x_i, x_j)$
- Where and how to cut dendrogram (D^*)

More on cutting dendrogram

- For visualization, smaller # clusters is preferable
- For tasks like outlier detection, cut based on:
  - Distance threshold
  - Inconsistency coefficient
    - Compare height of merge to average merge heights below

More on cutting dendrogram (cont'd)

- If top merge is substantially higher, then it is joining two subsets that are relatively far apart compared to the members of each subset internally
- Still have to choose a threshold to cut at, but now in terms of inconsistency rather than distance

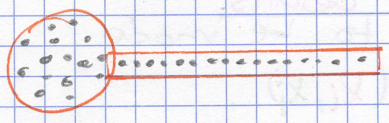
- No cutting method is "incorrect", some are just more useful than others

Computational considerations

- Computing all pairs of distances is expensive
  - Brute force algorithm is $O(N^2 \log N)$ (N: #datapoints)
- Smart implementations use triangle inequality to rule out candidate pairs
- Best known algorithm is $O(N^2)$

Statistical issues

Chaining: Distant points clustered together if there is a chain of pairwise close points between



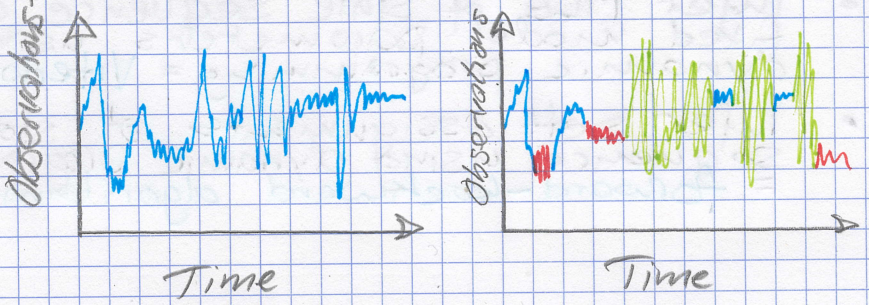
Other linkage functions can be more robust, but restrict the shapes of clusters that can be found

- Complete linkage
  - max pairwise distance between clusters
- Ward criterion
  - min within-cluster variance at each merge

Hidden Markov models

Hidden Markov models (HMMs): Another notion of "clustering"

What if we have time series data?



Would be hard to distinguish red, blue, and green clusters if we ignored order of data

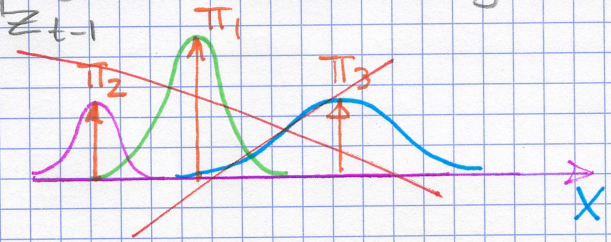
Hidden Markov model (HMM)

As in mixture model...

Every observation X_t is associated with cluster assignment variable Z_t

Each cluster has a distribution over observed values

Difference from mixture model:
Probability of $(Z_t = k)$ depends on previous cluster assignment



Inference in HMMs

- Learn MLE of HMM parameters using EM algorithm = Baum Welch
- Infer MLE of state sequence given fixed model parameters using dynamic programming = Viterbi algorithm
- Infer soft assignments of state sequence using dynamic programming = forward-backward algorithm