Unsupervised Statistical Learning: Gaussian Mixture Models

Siva Balakrishnan
Data Mining: 36-462/36-662

March 28th, 2019

Project Updates

1. Release date: First week of April.
2. Deliverables:
   - **Predictions due:** April 30th.
   - **Final report due:** May 3rd.
3. Teams: You can do projects individually or in teams of 2 (preferable). If you want me to (randomly) pair you with another person use this google doc. I will do this on April 5th at midnight (after the release of the project).

`https://docs.google.com/document/d/1HkWV112R8XR1mCDBJ6RHlxACRY25PB7zwzbj4tbDt_g/edit?usp=sharing`

Recap: K-medoids

- Just like K-means except we want the centers $c_1, \ldots, c_K$ to be actual data points.
- Initial guess for centers $c_1, \ldots c_K$ (e.g., randomly select K of the points $X_1, \ldots X_n$), then repeat:
  1. Minimize over C: for each $i = 1, \ldots n$, find the cluster center c_k closest to X_i, and let $C(i) = k$
  2. Minimize over $c_1, \ldots c_K$: for each $k = 1, \ldots K$, let $c_k = X_k^*$, the medoid of points in cluster k, i.e., the point X_i in cluster k that minimizes $\underline{\sum_{C(j)=k} \|X_j - X_i\|_2^2}$

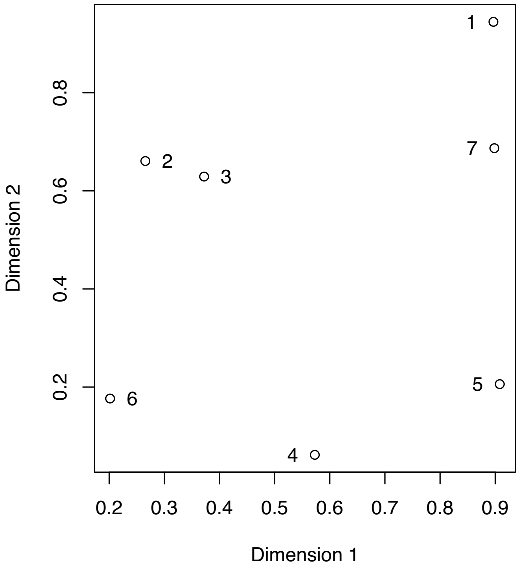
  Stop when within-cluster variation doesn't change
- Advantages over K-means: $c_1, \ldots, c_K$ interpretable
- Disadvantages relative to K-means:

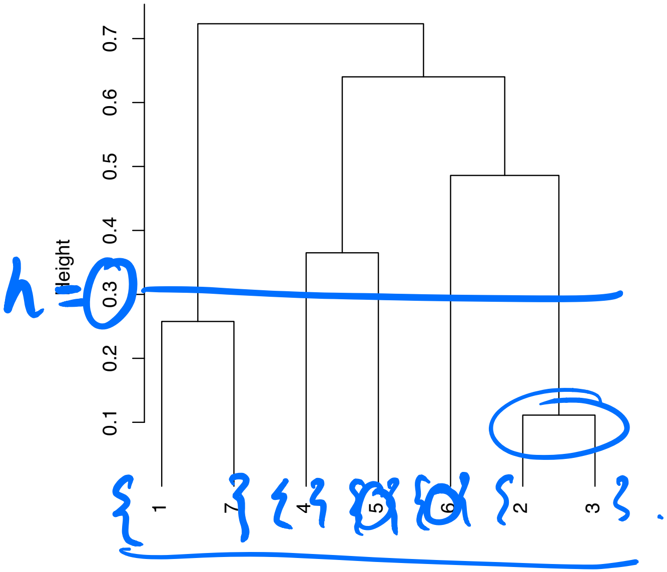
→ k-means cost will be worse

→ computing medoids is harder.

Recap: Hierarchical Clustering

- Want to produce a sequence of *nested* clusters. No need to specify K anymore.
- Two broad strategies: agglomerative and divisive. → recursive partitioning.
- Represent hierarchical clusters by a dendrogram: cut horizontally to get clusters, and heights tell us about (dis)similarities, groups that merge near the bottom are quite similar.





Recap: Linkage

- To specify an agglomerative hierarchical clustering algorithm we *only specify one thing*: the linkage rule. This is just a way to assign a distance to two *groups* of points (usually derived from a distance between individual points).
- The most canonical ways to do this:
  1. Single Linkage:

$$d_{\text{single}}(G,H) = \min_{\boxed{i\in G},\, \boxed{j\in H}} \boxed{d_{ij}}$$

  2. Complete Linkage:

$$d_{\text{complete}}(G,H) = \underline{\max_{i\in G,\, j\in H} d_{ij}}$$

  3. Average Linkage:

$$d_{\overset{\text{average}}{\cancel{\text{complete}}}}(G,H) = \frac{1}{n_G n_H} \sum_{i\in G,\, j\in H} d_{ij} \Big\} .$$

Recap: Cut Interpretations

separation guarantee.

- Suppose we cut a (single/complete/average)-linkage dendrogram at some height h to get clusters. Can we say anything nice about the clusters?

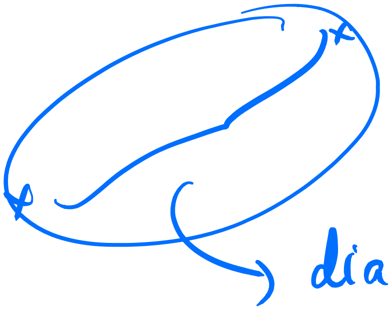
- **Single Linkage:** → if x is in C & there is at least one other pt then $\exists y$ $d_{xy} \leq h$.

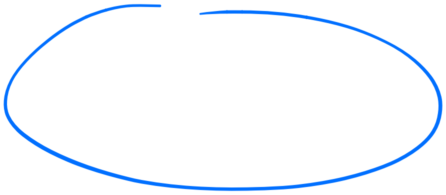
  $x \in C_1, y \in C_2$ $d_{xy} > h$.

- **Complete Linkage:** every pair $x, y \in C$ $d_{xy} \leq h$.

  diameter → guarantee

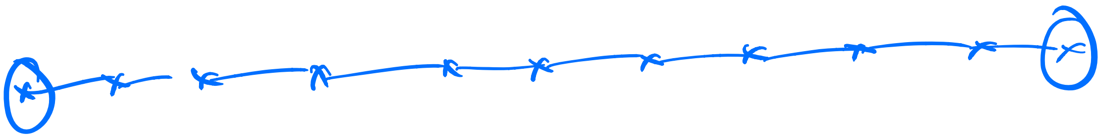
- **Average Linkage:** Nothing particularly interesting to say here.





diameter of cluster =
dist b/w farthest pts.

Single



Common properties

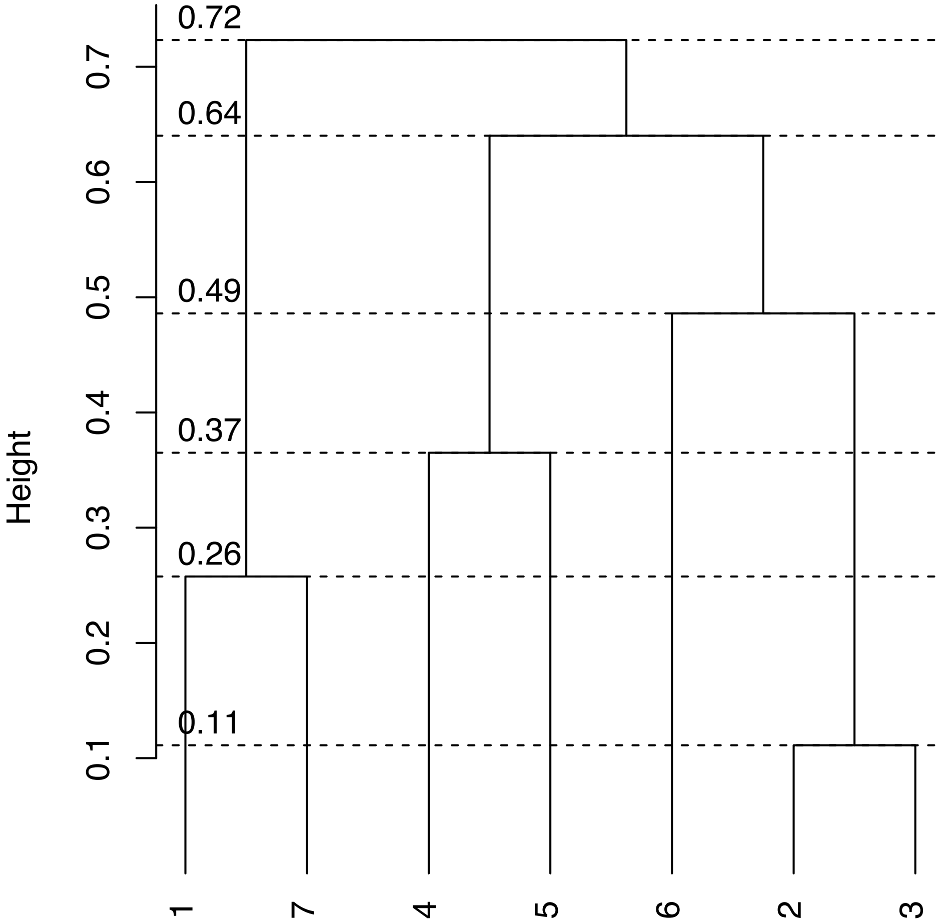
Single, complete, average linkage share the following properties:

- These linkages operate on dissimilarities d_{ij}, and don't need the points $X_1, \ldots X_n$ to be in Euclidean space
- Running agglomerative clustering with any of these linkages produces a dendrogram with no inversions

Second property, in words: disimilarity scores between merged clusters only increases as we run the algorithm

Means that we can draw a proper dendrogram, where the height of a parent is always higher than height of its daughters

Example of a dendrogram with no inversions



Shortcomings of single, complete linkage

clusters may not be compact.

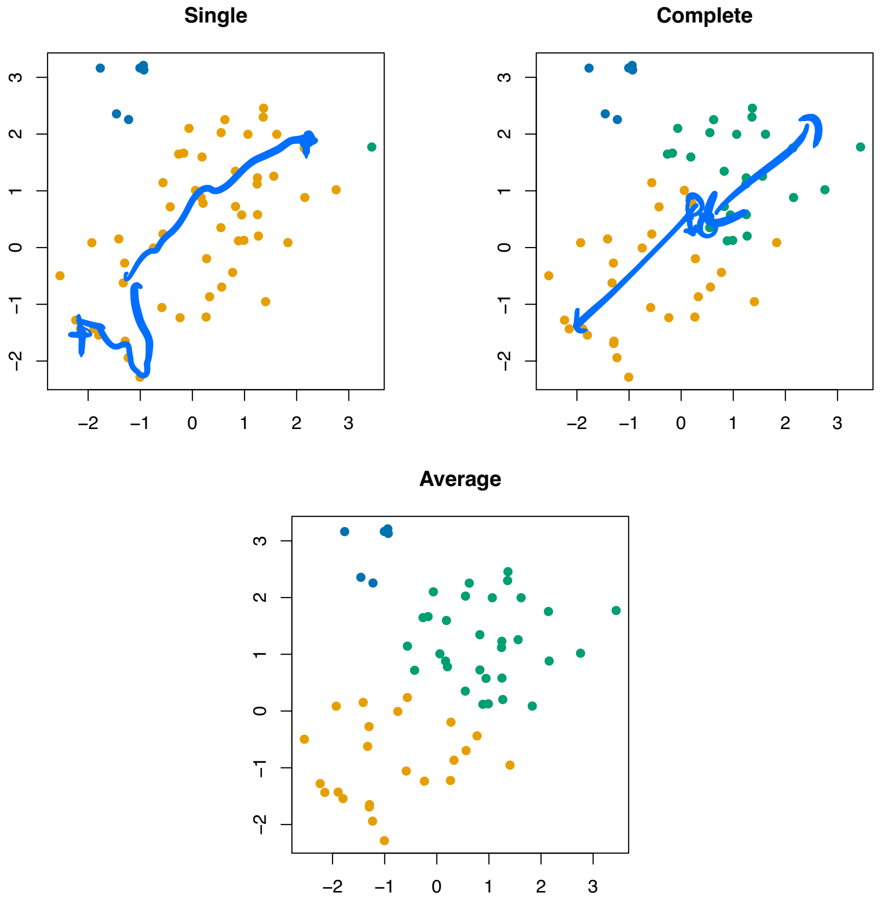
Single and complete linkage can have some practical problems:

- Single linkage suffers from chaining. In order to merge two groups, only need one pair of points to be close, irrespective of all others. Therefore clusters can be too spread out, and not compact enough
- Complete linkage avoids chaining, but suffers from crowding. Because its score is based on the worst-case dissimilarity between pairs, a point can be closer to points in other clusters than to points in its own cluster. Clusters are compact, but not far enough apart

clusters may not be well-sep

Average linkage tries to strike a balance. It uses average pairwise dissimilarity, so clusters tend to be relatively compact and relatively far apart

Example of chaining and crowding



Single

Complete

Average

Shortcomings of average linkage

Average linkage isn't perfect, it has its own problems:

- It is not clear what properties the resulting clusters have when we cut an average linkage tree at given height h. Single and complete linkage trees each had simple interpretations
- Results of average linkage clustering can change with a monotone increasing transformation of dissimilarities d_{ij}. I.e., if h is such that $h(x) \leq h(y)$ whenever $x \leq y$, and if $x < y$ then $h(x) < h(y)$, and we used dissimilarites $h(d_{ij})$ instead of d_{ij}, then we could get different answers

$\log d_{ij}$

d_{ij}

d_{ij}^2 → use

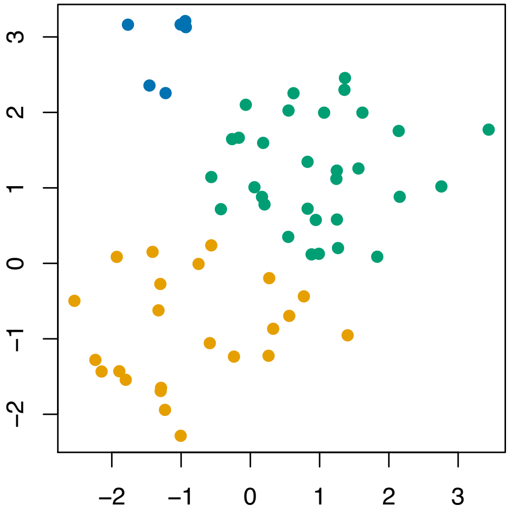
$\exp(d_{ij})$.

Depending on the context, second problem may be important or unimportant. E.g., it could be very clear what dissimilarities should be used, or not

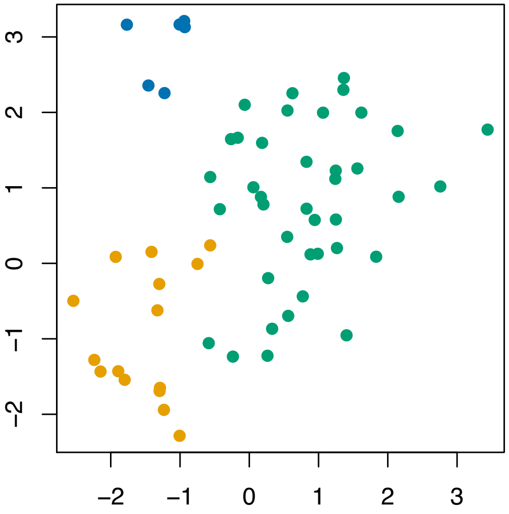
Note: results of single, complete linkage clustering are unchanged under monotone transformations

Example of a change with monotone increasing transformation

Avg linkage: distance



Avg linkage: distance^2



Recap: hierarchical agglomerative clustering

Hierarchical agglomerative clustering: start with all data points in their own groups, and repeatedly merge groups, based on linkage function. Stop when points are in one group (this is agglomerative; there is also divisive)

This produces a sequence of clustering assignments, visualized by a dendrogram (i.e., a tree). Each node in the tree represents a group, and its height is proportional to the dissimilarity of its daughters

Three most common linkage functions: single, complete, average linkage. Single linkage measures the least dissimilar pair between groups, complete linkage measures the most dissimilar pair, average linkage measures the average dissimilarity over all pairs

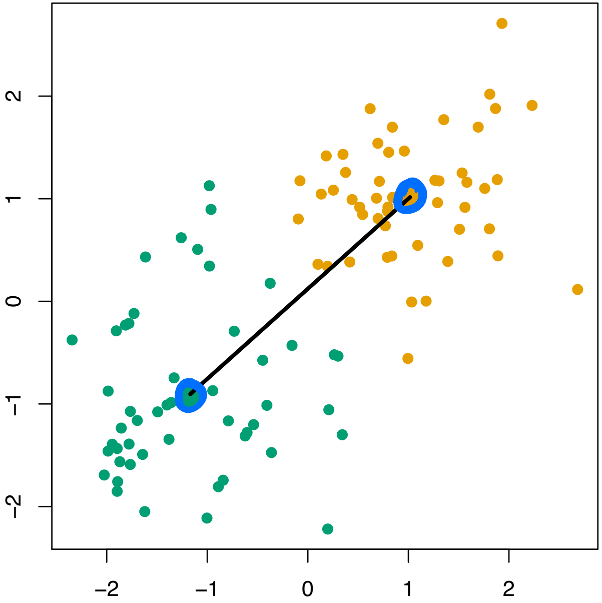
Each linkage has its strengths and weaknesses

Centroid linkage

Centroid linkage[1] is commonly used. Assume that $X_i \in \mathbb{R}^p$, and $d_{ij} = \|X_i - X_j\|_2$. Let $\bar{X}_G, \bar{X}_H$ denote group averages for G, H. Then:

$$d_{\text{centroid}}(G,H) = \|\underline{\bar{X}_G} - \underline{\bar{X}_H}\|_2 \rightarrow$$

Example (dissimilarities d_{ij} are distances, groups are marked by colors): centroid linkage score $d_{\text{centroid}}(G,H)$ is the distance between the group centroids (i.e., group averages)

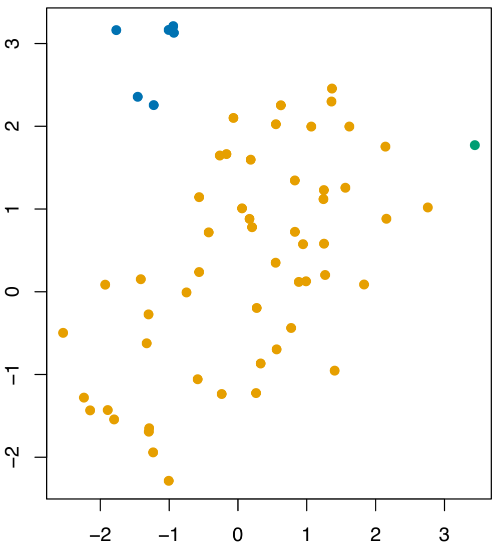


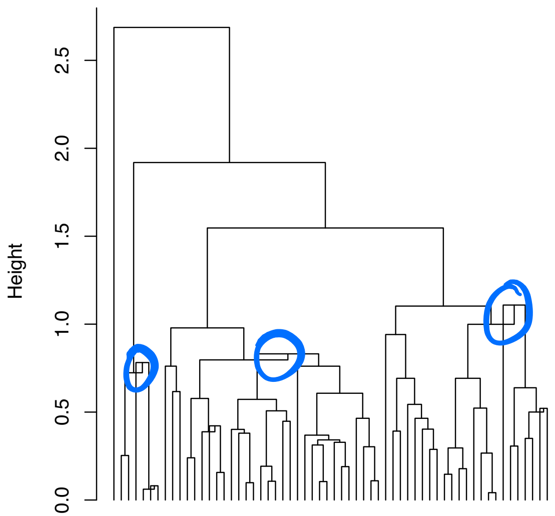
[1] Eisen et al. (1998), "Cluster Analysis and Display of Genome-Wide Expression Patterns"

Centroid linkage is the standard in biology

Centroid linkage is simple: easy to understand, and easy to implement. Maybe for these reasons, it has become the standard for hierarchical clustering in biology

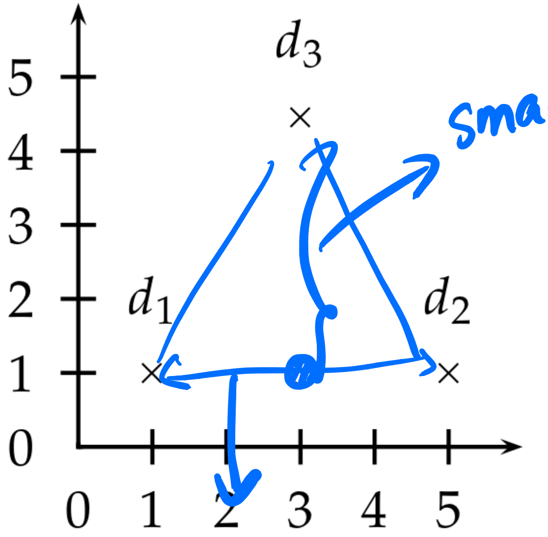
Here $n = 60$, $X_i \in \mathbb{R}^2$, $d_{ij} = \|X_i - X_j\|_2$. Cutting the tree at some heights wouldn't make sense ... because the dendrogram has inversions! But we can, e.g., still look at output with 3 clusters



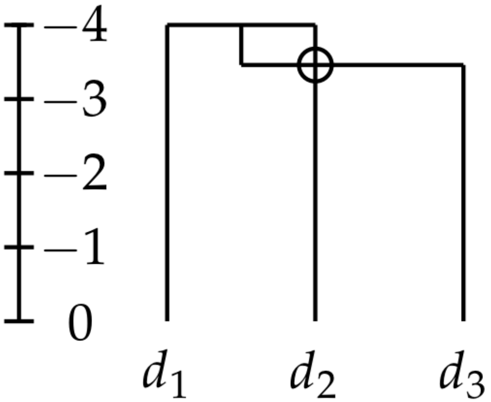


Cut interpretation: there isn't one, even with no inversions

Centroid Linkage Inversions



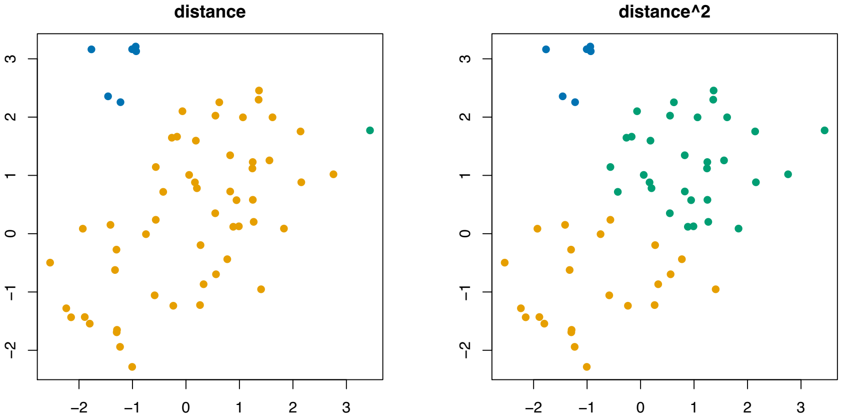
smaller than other distances.



Slightly closer.

Shortcomings of centroid linkage

- Can produce dendrograms with inversions, which really messes up the visualization
- Even if were we lucky enough to have no inversions, still no interpretation for the clusters resulting from cutting the tree
- Answers change with a monotone transformation of the dissimilarity measure $d_{ij} = \|X_i - X_j\|_2$. E.g., changing to $d_{ij} = \|X_i - X_j\|_2^2$ would give a different clustering



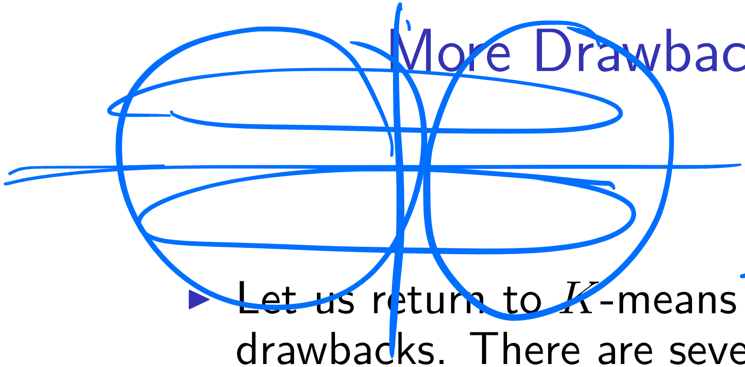
Linkages summary

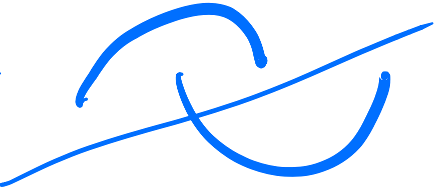
Linkage	No inversions?	Unchanged with monotone transformation?	Cut interpretation?	Notes
Single	✓	✓	✓	chaining
Complete	✓	✓	✓	crowding
Average	✓	×	×	
Centroid	×	×	×	simple

Note: this doesn't tell us what "best linkage" is.

Remember that choosing a linkage can be very situation dependent.

More Drawbacks and More Ideas

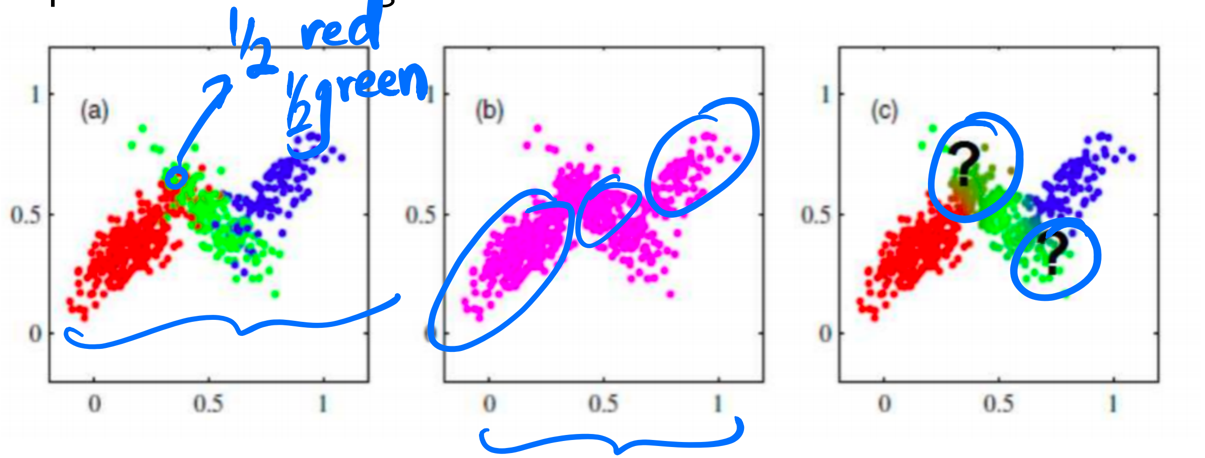




- Let us return to K-means and attempt to address more of its drawbacks. There are several that we listed and discussed before but two are useful to keep in mind for now:
  - K-means is not good at finding non-spherical clusters.
  - K-means (and the hierarchical methods) is a “hard” clustering method, i.e. each point gets assigned to exactly one cluster. We might have *overlapping* clusters and K-means would not be ideal for this setting.
- As an aside, none of the clustering methods we have seen so far are particularly statistical.

A Picture

- We can see overlapping clusters.
- Points that could reasonably belong to either group are impossible to distinguish.



- Suppose we denote by z_i the (unobserved) cluster membership for the i-th point. Just like in classification maybe we want to model/estimate:

$$P(z_i = \text{red}|x_i), P(z_i = \text{green}|x_i), \text{and so on.}$$

These are probabilities of belonging to different clusters, i.e. they provide a *soft* clustering.

Density Estimation

- We want to estimate the density of the points in some way that allows us to extract clusters.

histograms.

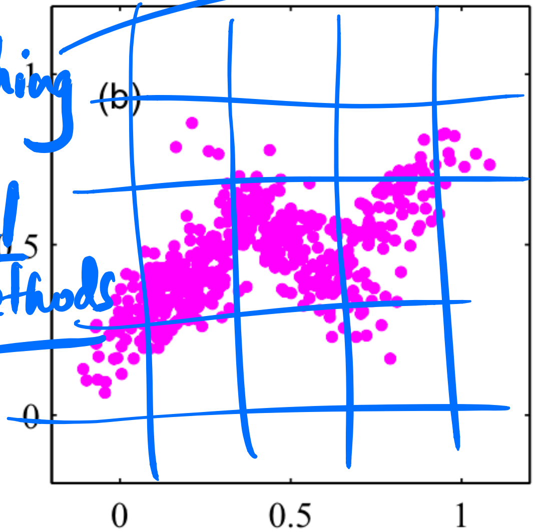
Parametric

Gaussian

Est. parameters

Smoothing

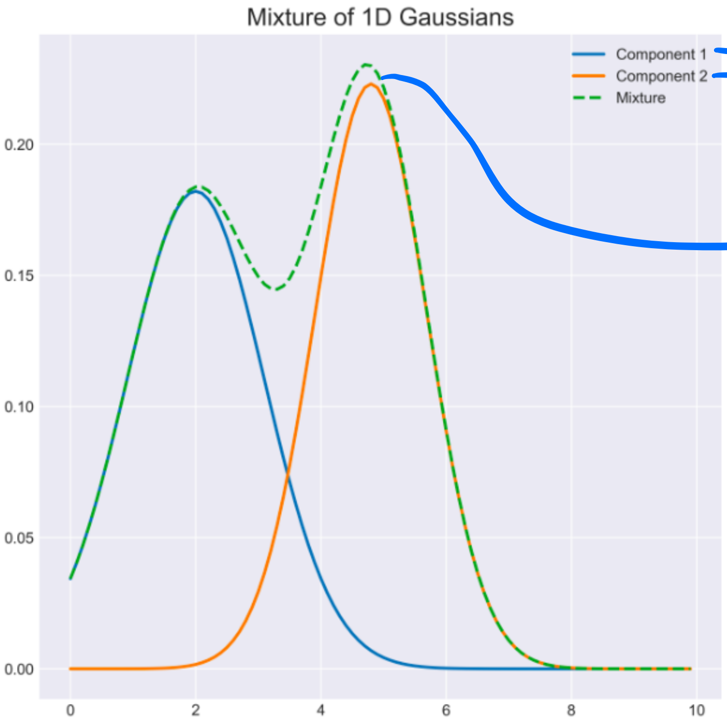
Local methods



- The density is clearly not Gaussian, and a kernel density estimator will not be directly useful for clustering. Want something in-between.

Mixture Models

- We are going to model the density as a *mixture* of simple distributions:



Gaussian

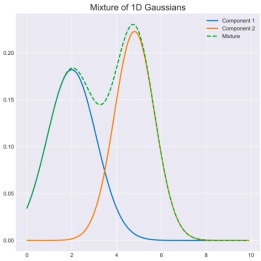
Gaussian

Bi-modal

"mixture".

Mixture Models

- We are going to model the density as a *mixture* of simple distributions:



“relative mass of each cluster”

weights.

- In particular,

$$\underbrace{\boxed{f(x)}}_{\text{also a distribution}} = \sum_{k=1}^{K} \overset{\text{weights.}}{\lambda_k} \underbrace{\boxed{f_k(x)}}_{\text{simple dist.}}.$$

where
  1. f_k are some simple distributions,
  2. $\lambda_k \geq 0$, $\sum_{k=1}^{K} \lambda_k = 1$, are called mixture weights.
  3. K is the number of mixture components (i.e. the number of clusters).

- Do not get confused – summing densities is not the same as summing draws from the distribution.

Mixture Models

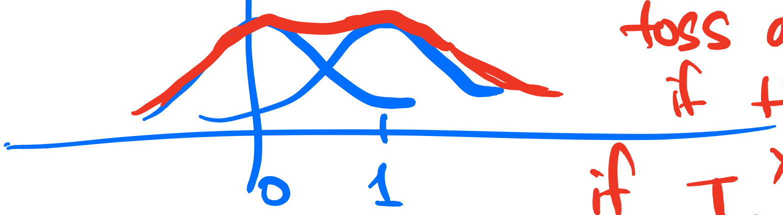
- How should we imagine mixture models? One way is to think about how to sample from them.

latent, cluster membership

- To sample from a mixture model:
  - First we draw $\underline{Z \in \{1, \dots, K\}}$, where $\mathbb{P}(Z = k) = \lambda_k$.
  - Then we draw a sample from the simple distribution f_Z.
  - Why does this work? We know that:

$$f(x) = \sum_{k=1}^{K} \mathbb{P}(Z = k) p(x | Z = k) = \sum_{k=1}^{K} \lambda_k f_Z(x).$$

- Again, to emphasize: $\boxed{X \sim \frac{1}{2} N(0, 1) + \frac{1}{2} N(1, 1)}$ is **not** the same as $X_1 \sim N(0, 1)$, $X_2 \sim N(1, 1)$, and $X = X_1 + X_2$.



toss a fair coin
if H then $X \sim N(0,1)$
if T, $X \sim N(1,1)$.

Gaussian Mixture Models

- By far the most popular mixture models are Gaussian Mixture Models (GMMs).
- In a GMM each simple distribution is a multivariate Gaussian. So we write:

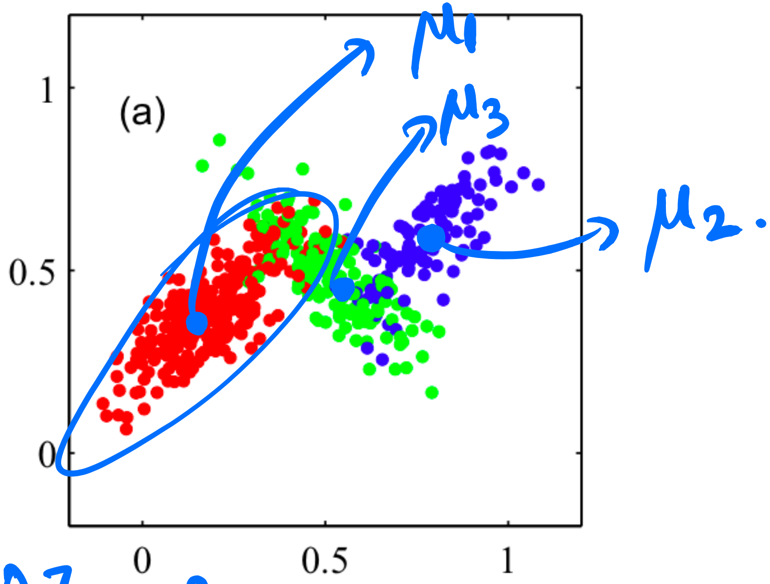
  $N(\mu_K, \Sigma_K).$

  $$f(x) = \sum_{k=1}^{K} \lambda_k \underline{\phi_k(x; \mu_k, \Sigma_k)},$$

  where $\phi_k(\cdot; \mu_k, \Sigma_k)$ denotes the Gaussian density with mean μ_k and covariance Σ_k.
- What are the parameters of this model?

  $\{\lambda_1 \ldots \lambda_K\}.$ $\{\mu_1, \ldots, \mu_K\}$ $\{\Sigma_1, \ldots, \Sigma_K\}.$

Gaussian Mixture Models



- Model: $\overset{0.7}{\lambda_1}, \overset{0.2}{\lambda_2}, \overset{0.1}{\lambda_3}$ and

$$f_{\text{red}}(x) = N(\mu_{\text{red}}, \Sigma_{\text{red}}),$$
$$f_{\text{blue}}(x) = N(\mu_{\text{blue}}, \Sigma_{\text{blue}}),$$
$$f_{\text{green}}(x) = N(\mu_{\text{green}}, \Sigma_{\text{green}}),$$

Mixture Models in Clustering

$$f(x) = \sum_{k=1}^{K} \lambda_k N(\mu_k, \Sigma_k).$$

- Suppose someone handed us a mixture model. How would we "soft" cluster our data?
- For a point x we would compute for $i \in \{1, \dots, K\}$:

$$N(x; \mu_i, \Sigma_i) = f_i(x) \qquad \lambda_i$$

$$\underline{P(Z = i | X = x)} = \frac{\overset{f_i(x)}{\mathbb{P}(X=x|Z=i)}\,\overset{\lambda_i}{\mathbb{P}(Z=i)}}{\sum_j \mathbb{P}(X=x|Z=j)\,\mathbb{P}(Z=j)}$$

- Does this expression make intuitive sense?

$$= \frac{\lambda_i f_i(x)}{\sum_{j=1}^{K} \lambda_j f_j(x)}.$$

Estimating a Mixture Model

- So we only have one real question remaining, given data $X_1, \dots, X_n$ how do we estimate the parameters of the mixture model i.e. the $(\lambda_k, \mu_k, \Sigma_k)$?
- Seems easy enough. We can use the LDA idea, take data from each group, compute the fraction of points (for λ_k), the mean (for μ_k) and the covariance (for Σ_k).

don't know labels.

Estimating a Mixture Model

- So we only have one real question remaining, given data $X_1, \ldots, X_n$ how do we estimate the parameters of the mixture model i.e. the $(\lambda_k, \mu_k, \Sigma_k)$?
- Seems easy enough. We can use the LDA idea, take data from each group, compute the fraction of points (for λ_k), the mean (for μ_k) and the covariance (for Σ_k).

- Any problems with this? Are the parameters of a mixture model even identifiable?
- In statistics we call these missing data or latent variable problems, i.e. the cluster memberships $Z_1, \ldots, Z_n$ are *missing*.

Estimating a Mixture Model – MLE

- If we observed $\{(X_1, Z_1), \dots, (X_n, Z_n)\}$ we would just use maximum likelihood, i.e. we would maximize:

log-likelihood

$$\ell\ell(\underline{(\lambda_k, \mu_k, \Sigma_k}_{\,})_{k=1}^{K}) = \underbrace{\sum_{i=1}^{n} \log f(X_i, Z_i)}_{},$$

  and this would give us the "LDA" estimates we discussed on the last slide.

- If we don't observe the Z_i's we should try to maximize the likelihood of the data we see (this is called the *marginal* likelihood):

observed data likelihood.

$$m\ell\ell((\lambda_k, \mu_k, \Sigma_k)_{k=1}^{K}) = \underline{\sum_{i=1}^{n} \log f(X_i)},$$

  equivalently:

$$m\ell\ell((\lambda_k, \mu_k, \Sigma_k)_{k=1}^{K}) = \sum_{i=1}^{n} \log \left[\sum_{k=1}^{K} f(X_i, k) \right],$$

  this is a hard problem in general.

Estimating a Mixture Model – Expectation-Maximization

- EM is a general method for (approximately) maximizing the (marginal) likelihood when you have missing data. We won't get too much into the details but describe the EM algorithm for GMMs directly.

→ analogous to assigning pts to clusters.

- Roughly, we want to first "guess" the latent variables Z_i and then if we knew those we could just maximize the (usual/complete) likelihood. → recomputing centroids.
- It resembles k-means. Except instead of assigning each point to a single cluster we "softly" assign them so they contribute fractionally to each cluster.

Estimating a Mixture Model – Expectation-Maximization

- We initialize the parameters $(\lambda_k, \mu_k, \Sigma_k)_{k=1}^K$ randomly, and then alternate the following two steps:

1. **E-step:** We compute the cluster memberships for each point i:

$$\lambda_k f_k / \sum_j \lambda_j f_j .$$

$$w_{ik} \qquad \leftarrow P(Z_i = k | X_i) = \frac{\lambda_k \phi_k(X_i; \mu_k, \Sigma_k)}{\sum_{j=1}^K \lambda_j \phi_j(X_i; \mu_j, \Sigma_j)}$$

$\downarrow$ weight for ith pt belonging to k-th cluster.

as before.

2. **M-step:** Recompute the parameters:

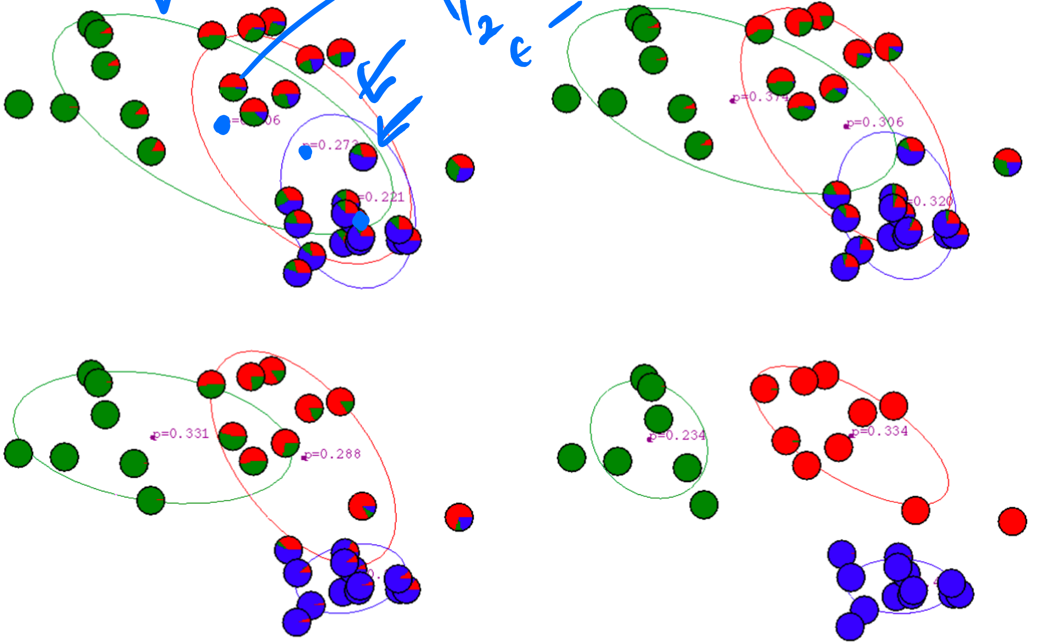
$$\lambda_k = \frac{\sum_{i=1}^n P(Z_i = k | X_i)}{n},$$

$$\mu_k = \frac{\frac{1}{n}\sum_{i=1}^{n} \hat{w}_{ik} X_i}{\frac{1}{n}\sum_{i=1}^{n} \hat{w}_{ik}} . \qquad \leftarrow \mu_k = \frac{\sum_{i=1}^n P(Z_i = k | X_i) X_i}{\sum_{i=1}^n P(Z_i = k | X_i)},$$

if no wts: just $\mu_k = \frac{1}{n_k} \sum_{i \in C_k} X_i$

and similarly update the covariance matrix.

Estimating a Mixture Model – EM in Action



$\frac{1}{2}$ – red
$\frac{1}{2} - \epsilon$ – green
ϵ – blue.