1.2 Variance Estimation for EM estimates

↗ if the likelihood is "well-behaved"

The EM algorithm finds the MLE, but it does not automatically produce an estimate of the covariance matrix. Why not?

Couldn't we just look at the curvature of the optimized surface?

No. We are optimizing Q, which is not the log-likelihood! ($Q - H$ is).

$\Rightarrow$ We cannot rely on information results without computing the Hessian (this may be hard).

There are several options to estimate the variance.

most straightforward

1. **Bootstrapping** (we will discuss bootstrapping in depth later in the course).

A simple bootstrap approach is easy to imagine:

1. Find $\hat{\theta}_{EM}$ from the original data $\underline{y} = \underline{y}_1, \ldots, \underline{y}_n$
2. Resample data w/ replacement to obtain bootstrap samples
   $\underline{y}^{*b} = (\underline{y}_1^{*b}, \ldots, \underline{y}_n^{*b})$ for $b = 1, \ldots, B$ ← # of bootstrap samples.
   Find $\hat{\theta}_{EM}^{*b}$ for each $\underline{y}^{*b}$
3. Use $\hat{\theta}_{EM}^{*b}$'s to get a CI for $\hat{\theta}_{EM}$ (percentile, basic, t-based, BCA, etc.)

Straightforward/easy.

Can be computationally expensive (need to run EM for $b = 1, \ldots, B$).

but embarassingly parallel, burden not too bad.

2. Louis's Method

$\ell(\underline{\theta}|\underline{y}) = Q(\underline{\theta},\underline{\theta}^{(v)}) - H(\underline{\theta},\underline{\theta}^{(v)}) \quad \forall\, \underline{\theta}^{(v)}$

So in particular, $\ell(\underline{\theta}|\underline{y}) = Q(\underline{\theta},\underline{\theta}) - H(\underline{\theta},\underline{\theta})$.

$\Rightarrow -\ell''(\underline{\theta}|\underline{y}) = -Q''(\underline{\theta},\underline{\theta}) + H''(\underline{\theta},\underline{\theta})$ derivatives on right side of $=$ wrt first argument.

$= n\,\overline{I}(\underline{y},\underline{\theta})$ from before.

observed information = "complete information" − "missing information".

Assuming integration and differentiation are interchangeable.

$-Q''(\underline{\theta},\underline{\theta}) = -E_{\underline{z}|\underline{y}}\left[\ell''(\underline{\theta}|\underline{y},\underline{z})\right] = -\int \ell''(\underline{\theta}|\underline{y},\underline{z})\, f_{\underline{z}|\underline{y}}(\underline{z}|\underline{y};\underline{\theta})\, d\underline{z}$

Somewhat similar in form to Fisher "information" except for the distribution expectation wrt. similar argument made about $H''(\underline{\theta},\underline{\theta})$.

So to get CI's, need $Q''(\underline{\theta},\underline{\theta})$ and $H''(\underline{\theta},\underline{\theta})$:

$Q''(\underline{\theta},\underline{\theta})$ is the curvature of the optimization surface $\Rightarrow$ can often get this numerically evaluated at $\hat{\theta}_{EM}$.

$$H''(\underline{\theta},\underline{\theta}) = \int \frac{\partial^2 \log f_{\underline{z}|\underline{y}}(\underline{z}|\underline{y},\underline{\theta})}{\partial \underline{\theta}^2} \underbrace{f_{\underline{z}|\underline{y}}(\underline{z}|\underline{y},\underline{\theta})}_{\text{same!}}\, d\underline{z} \quad \Rightarrow \text{using fact (2)}$$

$$= \mathrm{Var}\left[\frac{\partial \log f_{\underline{z}|\underline{y}}(\underline{z}|\underline{y},\underline{\theta})}{\partial \underline{\theta}}\right] \text{ wrt } f_{\underline{z}|\underline{y}}$$

A MC estimate of $H''(\underline{\theta},\underline{\theta})$ is thus the sample variance of $\dfrac{\partial \log f_{\underline{z}|\underline{y}}(\underline{z}|\underline{y};\underline{\theta})}{\partial \underline{\theta}}$ values of a sample of $\underline{z}$'s imputed.

Other options see Givens & Hoeting Section 4.2.3

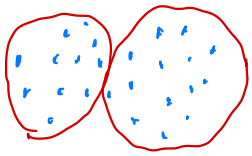
- SEM (supplemental)
- Empirical Information
- Numerical differentiation to get Hessian.

1.3 Another way to cluster: K-means

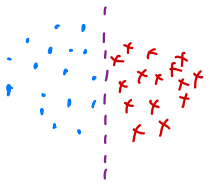
Goal of clustering:

Find an optimal grouping for which observations within each group are "similar" but clusters are "dissimilar" to each other.

Cluster analysis (unsupervised) vs. Classification (supervised)



goal: discover structure



goal: prediction

Methods for clustering include hierarchical and non-hierarchical, algorithmic and model-based.

hierarchical: proceed by going from $n \to 1$ (or $1 \to n$) clusters.

non hierarchical: model-based (finite mixture of Normals)

algorithmic (k-means)

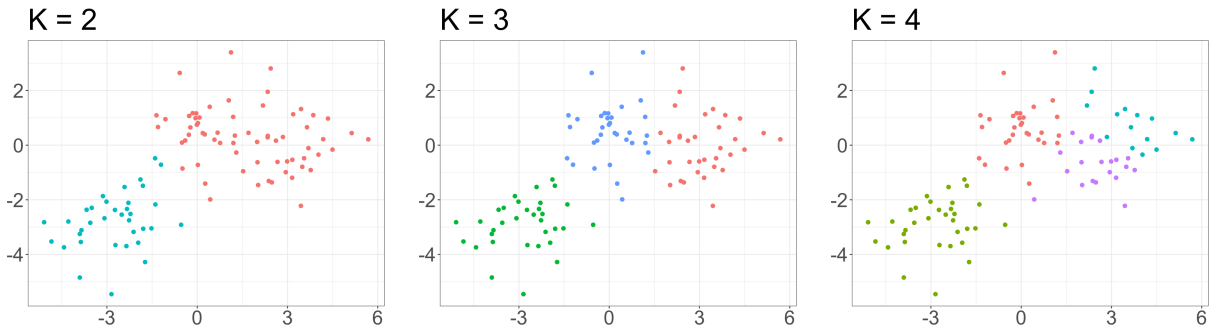
} both work for a set # of clusters but can be applied to different #s.

Not an exhaustive list.

K-means is a simple and elegant approach to partition a data set into K distinct, non-overlapping clusters.

First specify how many clusters (k), then k-means assigns each observation to one of the K clusters.

Eg clustering $n=100$ observations into K clusters using $p=2$ features.



The K-means clustering procedure results from a simple and intuitive mathematical problem. Let $C_1, \ldots, C_K$ denote sets containing the indices of observations in each cluster. These satisfy two properties:

e.g. if obs i is in cluster k, $i \in C_k$.

1. $C_1 \cup C_2 \cdots \cup C_k = \{1, \ldots, n\}$

   each observation belongs to one of the K clusters.

2. $C_k \cap C_{k'} = \emptyset \quad \forall\ k \neq k'$

   clusters are non overlapping.

Idea: Good clustering is one for which within-cluster variation is as small as possible.

↳ clustered observations are "similar"

The *within-cluster variation* for cluster C_k is a measure of the amount by which the observations within a cluster differ from each other. call $W(C_k)$.

Then we want to

$$\underset{C_1,\dots,C_K}{\text{minimize}} \left\{ \sum_{k=1}^{K} W(C_k) \right\}$$

i.e., partition data into K clusters s.t. total within-cluster variation is minimized.

To solve this, we need to define within-cluster variation.

(many ways).

Most common: squared Euclidean distance

$$W(C_k) = \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2$$

$|C_k|$: # points in C_k

This results in the following optimization problem that defines K-means clustering:

$$\underset{C_1,\dots,C_K}{\text{minimize}} \left\{ \sum_{k=1}^{K} \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2 \right\}$$

objective function.

This is very hard to solve exactly, $\approx K^n$ ways to partition n obs. into K clusters!

A very simple algorithm has been shown to find a local optimum to this problem:

↳ "pretty good solution"

1. Randomly assign a number from 1 to K to each observation (initialize cluster assignments).
2. Iterate until cluster assignments stop changing:
   (a) for each of the K clusters compute cluster centroid ← vector of the p feature means of obs. in each cluster.
   (b) assign each observation to closest centroid cluster
   ↖ euclidean distance

This algorithm is guaranteed to decrease value of the objective function at each step. When cluster assignments stop changing, this a local minimum → not necessarily global ⇒ clustering depends on initial random clusters!

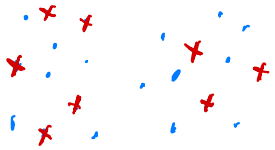
⇒ run the algorithm multiple times from different initial configurations and choose clustering w/ smallest objective function.

Still need K...

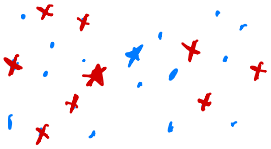
K-means illustration:

(1)

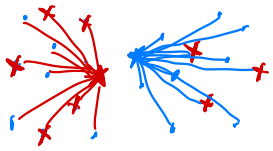
(2) Randomly assign



(3) Compute mediods



(4) Update assignments



repeat (3) & (4) until clusters are stable.

Questions about the algorithm:

1. How do we define distance?

   Normally Euclidean

   Could choose

   - Minkowski $d(\underline{x}_1, \underline{x}_2) = \left(\sum_{i=1}^{p} |x_{1i} - x_{2i}|^d \right)^{1/d}$
   - Mahalanobis

2. How do we choose starting values?

   randomly usually (shotgun approach)

   Maybe using another method? like hierarchical?

3. How do we choose K?

   look at between SS vs. within SS?

   Another approach: Dunn index compare to a "null" clustering

4. Should we scale the data?

   most times yes, unless no (LoL).

Compared to the Gaussian mixture problem,

↓ also fixed # of clusters.

In the Gaussian mixtures, we get probabilities of assignment ("soft" assignment), not just assignment ("hard" assignment).

↓ model-based (making assumptions!)

K-means sensitive to starting values.

In R, kmeans function will fit this algorithm