Clustering

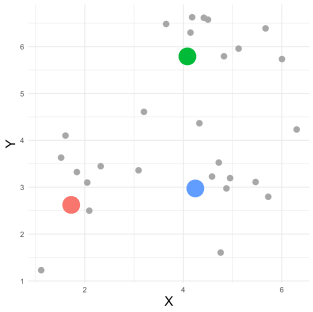
Lecture 12

Termeh Shafie

K-Means

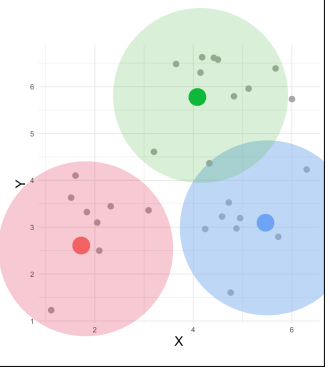
K-Means

1. Choose **k** random points as cluster centers



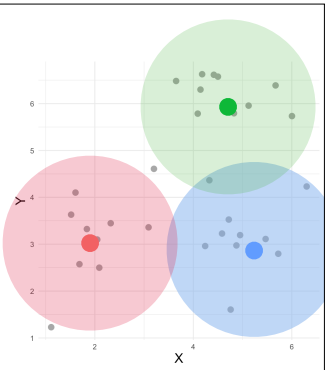
K-Means

1. Choose **k** random points as cluster centers
2. For each data point, assign it the cluster whose centroid is the closest



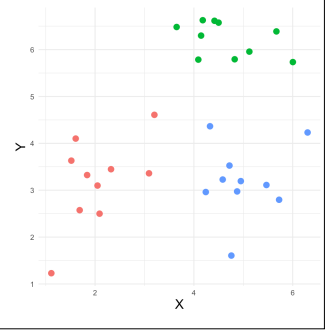
K-Means

1. Choose **k** random points as cluster centers
2. For each data point, assign it the cluster whose centroid is the closest
3. Using these assignments, recalculate the centers



K-Means

1. Choose **k** random points as cluster centers
2. For each data point, assign it the cluster whose centroid is the closest
3. Using these assignments, recalculate the centers
4. Reiterate from step (2) until **convergence**:
   - cluster membership does not change
   - center only changes very very little



K-Means

$$C_1 \cup C_2 \cup \ldots \cup C_K = \{1,\ldots,n\}$$

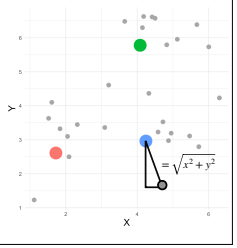
$$C_k \cap C_{k'} = \varnothing \text{ for all } k \neq k'$$

$$\min_{C_1,\ldots,C_K} \left\{ \sum_{k=1}^{K} W(C_k) \right\}$$

within cluster variance

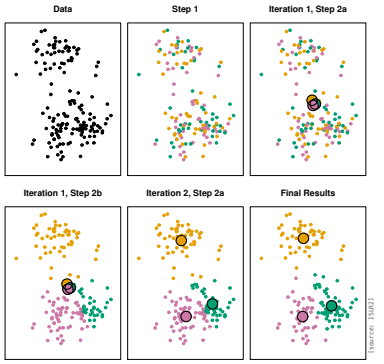
where $W(C_k) = \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2$

squared Euclidean distance



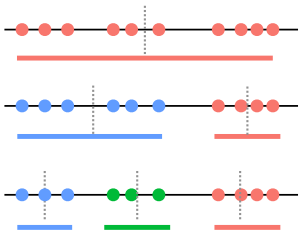
K-Means: Algorithm

1. Choose **k** random points as cluster centers
2. For each data point, assign it the cluster whose centroid is the closest
3. Using these assignments, recalculate the centers
4. Reiterate from step (2) until **convergence**:
   - cluster membership does not change
   - center only changes very very little



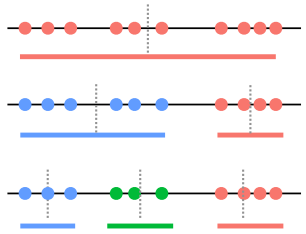
How to decide on K

- Start with K=1: compute total variation
- Then K=2: compute total variation
- Then K=3: compute total variation



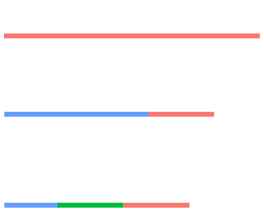
How to decide on K

- Start with K=1: compute total variation
- Then K=2: compute total variation
- Then K=3: compute total variation



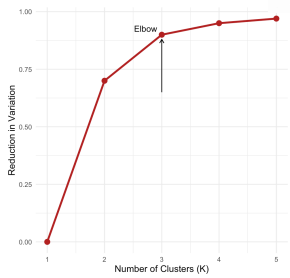
How to decide on K

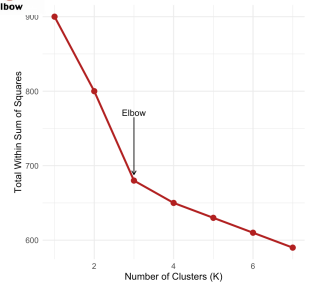
- Start with K=1: compute total variation
- Then K=2: compute total variation
- Then K=3: compute total variation



the reduction in total variation will get less and less as you increase K

How to decide on K





Distortion

metric that assesses the performance of K-means (smaller values better)

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \|x_n - \mu_k\|^2$$

actual data point n

center of cluster k

Goal: choose r_{nk} and μ_k that minimizes J

hard assignments!

$$r_{nk} = \begin{cases} 1 & \text{if } k = \arg\min_j \|x_n - \mu_j\|^2 \\ 0 & \text{otherwise} \end{cases}$$

$$\frac{dJ}{d\mu_k} = 2 \sum_{n=1}^{N} r_{nk}(x_n - \mu_k) = 0 \implies \mu_k = \frac{\sum_n r_{nk} x_n}{\sum_n r_{nk}} = \frac{1}{N_k} \sum_n r_{nk} x_n$$

optimal value for μ_k minimizing our loss is the mean of all data points in that cluster

Distortion

metric that assesses the performance of K-means (smaller values better)

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \|x_n - \mu_k\|^2$$

actual data point n

Goal: choose r_{nk} and μ_k that minimizes J

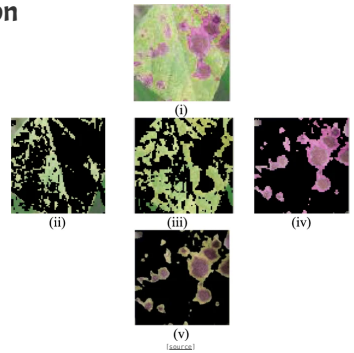
hard assignmen

$$\frac{dJ}{d\mu_k} = 2 \sum_{n=1}^{N} r_{nk}(x_n - \mu_k) = 0 \implies \mu_k = \frac{\sum_n r_{nk} x_n}{\sum_n r_{nk}} = \frac{1}{N_k} \sum_n r_{nk} x_n$$

optimal value for μ_k minimizing our loss is the mean of all data points in that cluster

1. choose **k** random points as cluster centers
2. for each data point, assign it the cluster whose centroid is the closest
3. using these assignments, recalculate the centers
4. reiterate from step (2) until **convergence**:
   - cluster membership does not change
   - center only changes very very little

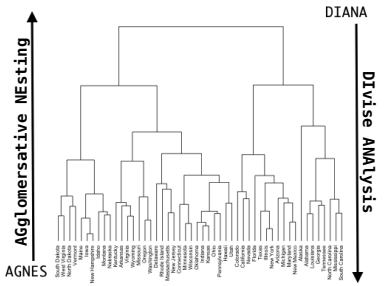
Application



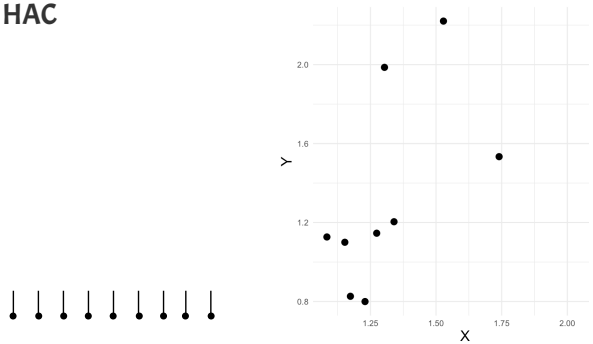
[source]

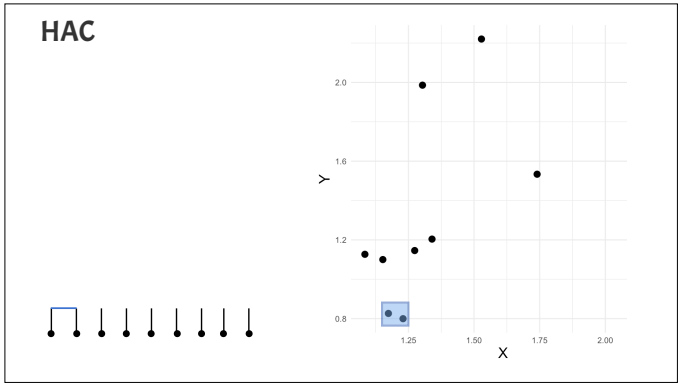
Hierarchical Agglomerative Clustering (HAC)

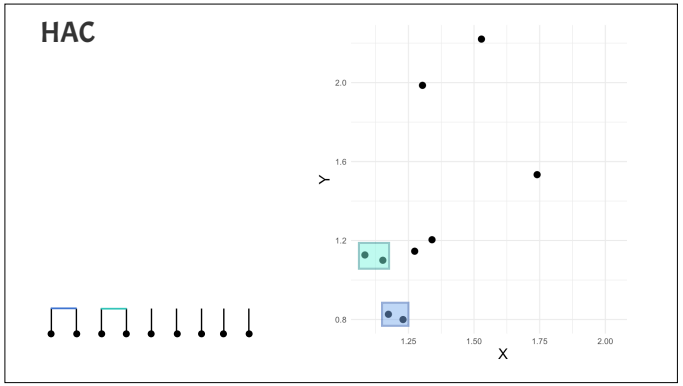
AGNES/DIANA

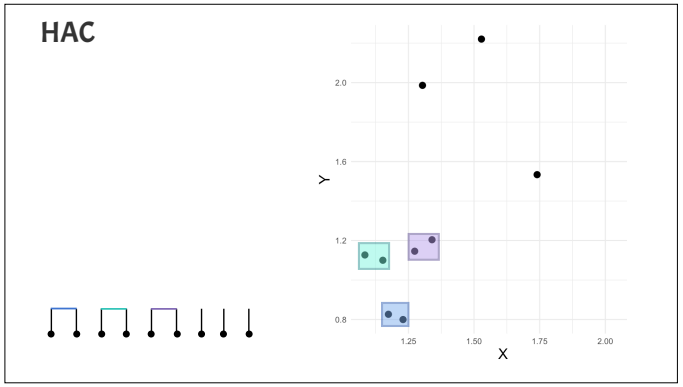


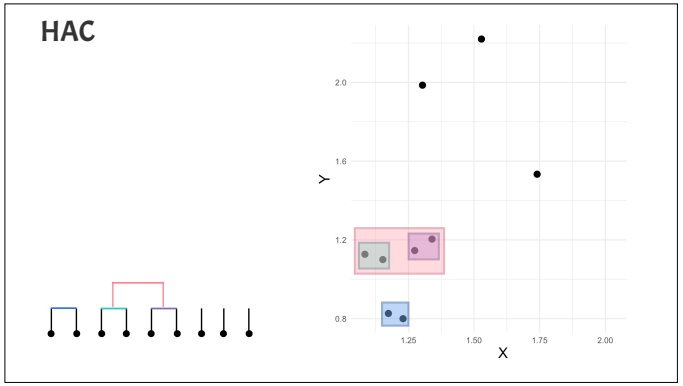
HAC

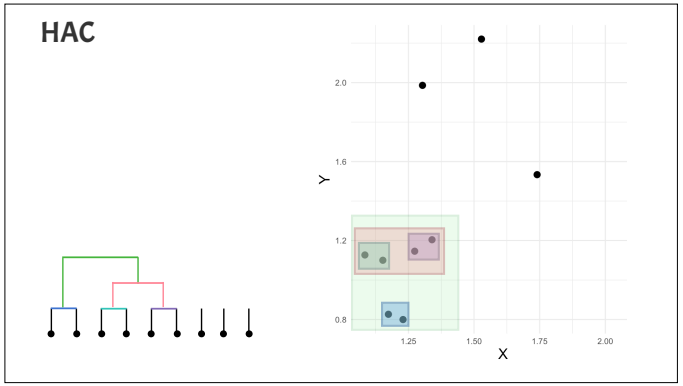


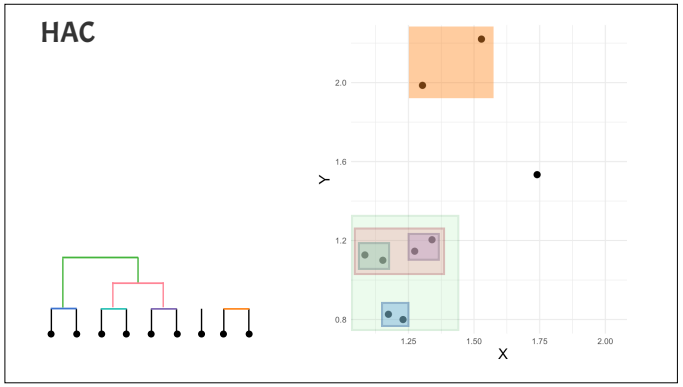

HAC
2.0
1.6
1.2
0.8
Y
1.25
1.50
1.75
2.00
X


HAC
2.0
1.6
1.2
0.8
Y
1.25
1.50
1.75
2.00
X

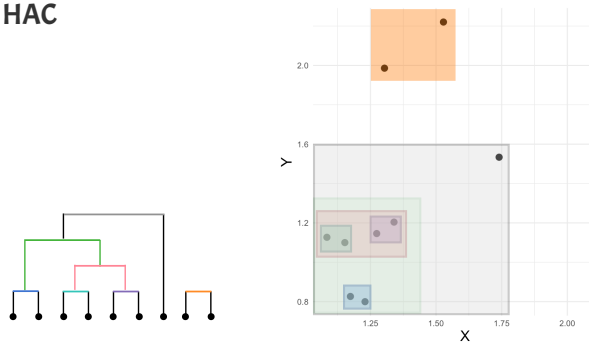

HAC
2.0
1.6
1.2
0.8
Y
1.25
1.50
1.75
2.00
X


HAC
2.0
1.6
1.2
0.8
Y
1.25
1.50
1.75
2.00
X

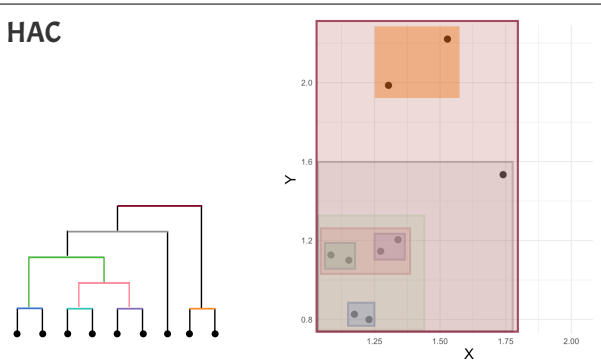

HAC
2.0
1.6
1.2
0.8
Y
1.25
1.50
1.75
2.00
X


HAC
2.0
1.6
1.2
0.8
Y
1.25
1.50
1.75
2.00
X

HAC

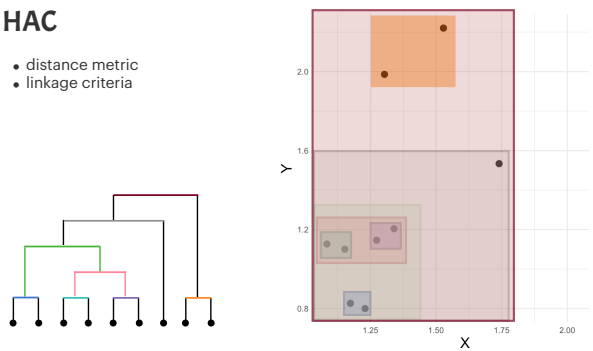


HAC



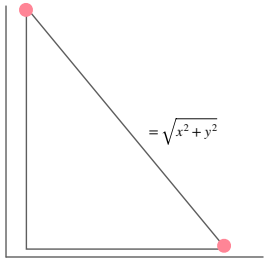
HAC

- distance metric
- linkage criteria



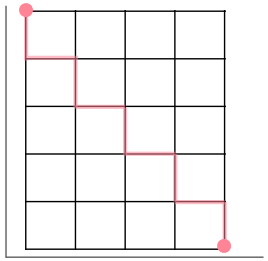
Distance Metrics: Euclidean

continuous data



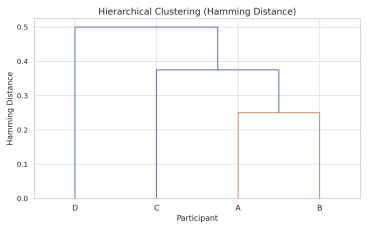
Distance Metrics: Manhattan

continuous data and high dimensions
(but also discrete or binary attributes)



Distance Metrics: Hamming

binary and categorical data

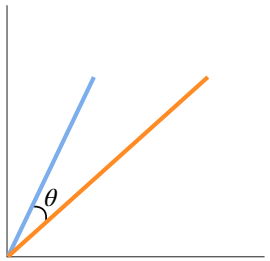


Participant	Q1	Q2	Q3	Q4
A	1	0	1	0
B	1	1	1	0
C	0	0	1	0
D	1	1	0	0

	A	B	C	D
A	0	1	2	3
B	1	0	3	1
C	2	3	0	3
D	2	1	3	0

Distance Metrics: Cosine

measures the angular difference between two vectors in a multi-dimensional space
word (or other) count data

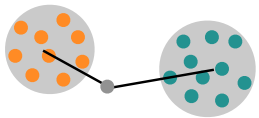


Linkage Criteria

how compare different clusters?

Linkage Criteria

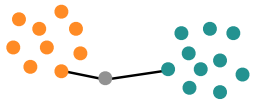
how compare different clusters?



the average of each clusters (“centroid”)

Linkage Criteria

how compare different clusters?



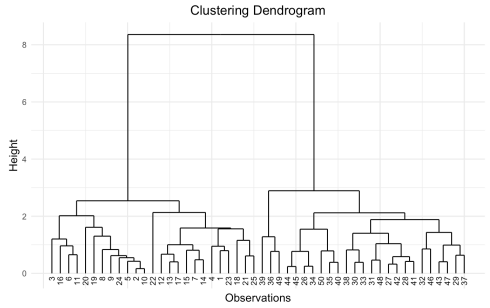
the closest point in each cluster ("single linkage")

Linkage Criteria

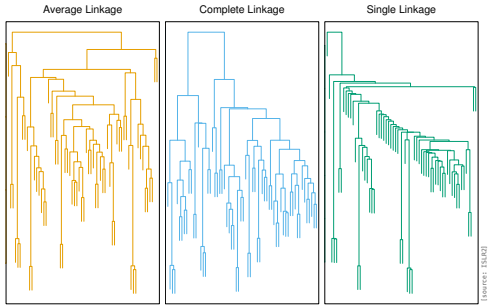
how compare different clusters?

the furthest point in each cluster ("complete linkage")

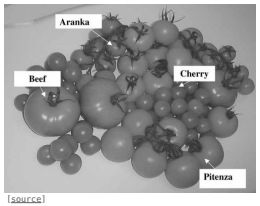
Reading a Dendrogram



Balanced Clusters and Density



Applications



[source]

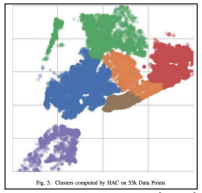
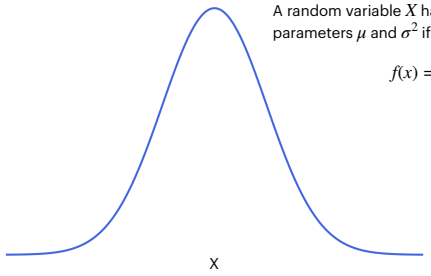


Fig. 5. Clusters computed by HAC on 55k Data Points

[source]

Gaussian Mixture Models (GMMs)

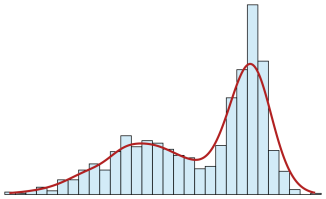
Normal (Gaussian) Distribution

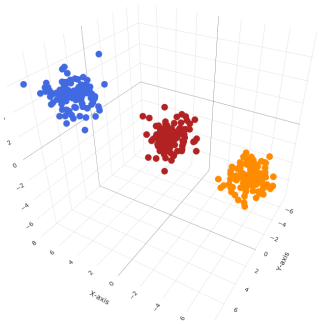


A random variable X has normal distribution with parameters μ and σ^2 if it has the following pdf:

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Multimodal Distribution





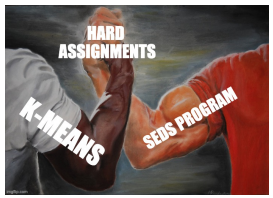
GMM

K-Means

- Hard assignment
- All variances are the same
- Roughly the same number of data points

GMM

- Soft (probabilistic) assignment
- Variances can be different
- Explicitly models number of data points



Recall K-Means Algorithm

metric that assesses the performance of K-means (smaller values better)

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \|x_n - \mu_k\|^2$$

actual data point n

minimizes J

hard assignment

1. Choose **k** random points as cluster centers
2. For each data point, assign it the cluster whose centroid is the closest
3. Using these assignments, recalculate the centers
4. Reiterate from step (2) until **convergence**:
   - cluster membership does not change
   - center only changes very very little

$$\frac{dJ}{d\mu_k} = 2\sum_{n=1} r_{nk}(x_n - \mu_k) = 0 \longrightarrow \mu_k = \frac{\sum_n r_{nk} x_n}{\sum_n r_{nk}} = \frac{1}{N_k}\sum_n r_{nk} x_n$$

optimal value for μ_k minimizing our loss is the mean of all data points in that cluster

GMM: EM Algorithm

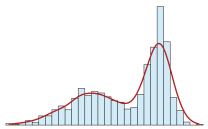
1. Choose **k** random points to be cluster centers (or estimate using k-means...)
2. For each data point, calculate the **probability** of belonging to each cluster
3. Using these probability weights, recalculate the **means + variances** (and weights)
4. Repeat 2 and 3 until **distributions converge**

Multimodal Distribution

$$p(x) = \sum_{k=1}^{K} w_k \, \mathcal{N}(x \mid \mu_k, \Sigma_k)$$

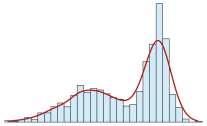
probability of being in group k

likelihood of seeing x in group k



Posterior Probabilities

$$p(x) = w_k \, \mathcal{N}(x \mid \mu_k, \Sigma_k)$$



prior probability of being in group k

likelihood of seeing x in group k

$$p(\text{cluster } k \mid x) = \frac{w_k \, \mathcal{N}(x \mid \mu_k, \Sigma_k)}{\sum_{j=1}^{K} w_j \, \mathcal{N}(x \mid \mu_j, \Sigma_j)}$$

posterior probability of being in cluster k

MaximumLikelihood Estimation

$$p(x) = \sum_{k=1}^{K} w_k \, \mathcal{N}(x \mid \mu_k, \Sigma_k)$$

$$p(\mathbf{X} \mid \mathbf{w}, \mu, \mathbf{\Sigma}) = p(x_1, x_2, \ldots, x_n \mid \mathbf{w}, \mu, \mathbf{\Sigma}) = \prod_{n=1}^{N} \sum_{k=1}^{K} w_k \mathcal{N}(x_n \mid \mu_k, \Sigma_k)$$

$$\log p(\mathbf{X} \mid \mathbf{w}, \mu, \mathbf{\Sigma}) = \sum_{n=1}^{N} \log \left[\sum_{k=1}^{K} w_k \mathcal{N}(x_n \mid \mu_k, \Sigma_k) \right]$$

Goal: choose w, μ, Σ that maximizes the log likelihood

GMM: EM Algorithm

1. Choose **k** random points to be cluster centers (or estimate using k-means...)
2. For each data point, calculate the **probability** of belonging to each cluster
3. Using these probability weights, recalculate the **means + variances** (and weights)
4. Repeat 2 and 3 until **distributions converge**

The E-step in EM Algorithm

prior probability of being in group k | likelihood of seeing x in group k

$$p(\text{cluster } k \mid x) = \frac{w_k \, \mathcal{N}(x \mid \mu_k, \Sigma_k)}{\sum_{j=1}^{K} w_j \, \mathcal{N}(x \mid \mu_j, \Sigma_j)}$$

posterior probability of being in cluster k

Responsibilities are the posterior probability of a data point being in cluster

How likely is the cluster? Many/few data points there?

How well does that data fit with that cluster?

$$r_{nk} = \frac{w_k \, \mathcal{N}(x \mid \mu_k, \Sigma_k)}{\sum_{j=1}^{K} w_j \, \mathcal{N}(x \mid \mu_j, \Sigma_j)}$$

normalize to get a probability

Responsibility is high if the data point is likely to belong to that cluster rather than other clusters

this is soft assignment

GMM: EM Algorithm

1. Choose **k** random points to be cluster centers (or estimate using k-means...)
2. For each data point, calculate the **probability** of belonging to each cluster
3. Using these probability weights, recalculate the **means + variances** (and weights)
4. Repeat 2 and 3 until **distributions converge**

The M-step in EM Algorithm

Via MLE we get the following estimates:

$$\mu_k = \frac{1}{N_k} \sum_{n=1}^{N} r_{nk} x_n \qquad \Sigma_k = \frac{1}{N_k} \sum_{n=1}^{N} r_{nk} (x_n - \mu_k)(x_n - \mu_k)^T \qquad w_k = \sum_{n=1}^{N} r_{nk}$$

the higher the responsibility of a data point for a cluster is, the more influence it has on what the mean and variance is

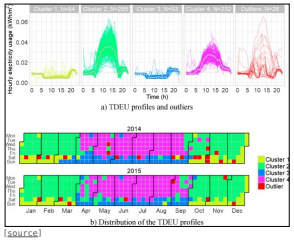
Note: $N_k = \sum_{n=1}^{N} r_{nk}$ is now based on soft assignments now

if data points are unlikely to belong to cluster k, the N_k small, if data points are likely to belong to cluster k, then N_k large

Take Aways

- GMM does **soft assignment**, every data point belongs to every cluster with some probability
- Data points that are more likely to be in a cluster have **more influence** over its parameters
- GMM uses the EM algorithm to iteratively update the cluster distributions:
  - first assign a responsibility to each data point (**E-step**)
  - then using them to calculate weighted means and variances for each cluster (**M-step**)
- Responsibilities measure **the probability of a data point being in each cluster** (technically the posterior probability).
- Responsibilities contain information about how common a cluster is as well as **the likelihood of a data point belonging to that cluster**

Applications



[source]

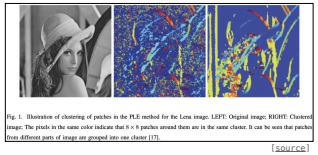
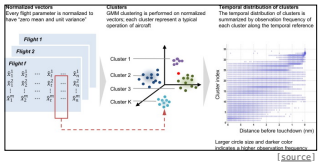


Fig. 1. Illustration of clustering of patches in the PLE method for the Lena image. LEFT: Original image; RIGHT: Clustered image. The pixels in the same color indicate that 8 × 8 patches around them are in the same cluster. It can be seen that patches from different parts of image are grouped into one cluster [17].

[source]

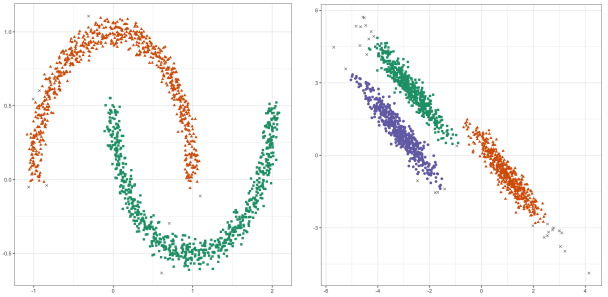


[source]

DBSCAN

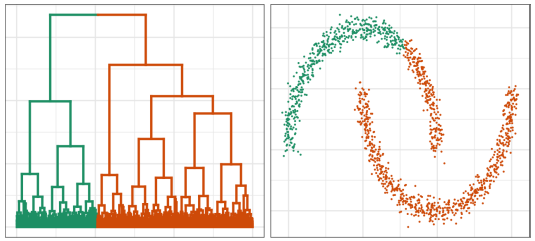
Density **B**ased **S**patial **C**lustering of **A**pplications with **N**oise

Example Data Structures



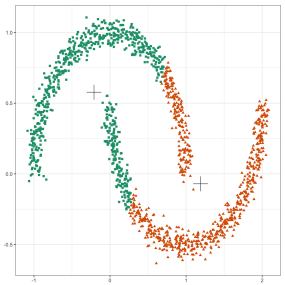
Where Others Fail...

HAC

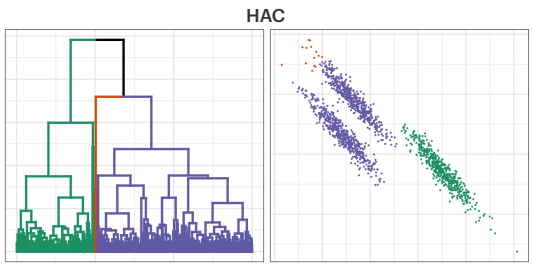


Where Others Fail...

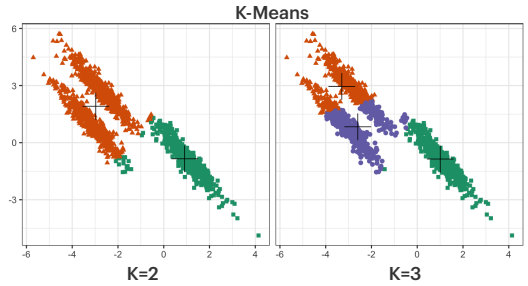
K-Means



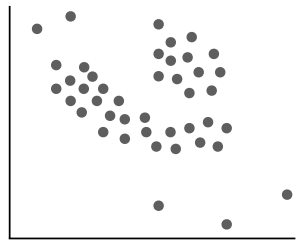
Where Others Fail...



Where Others Fail...

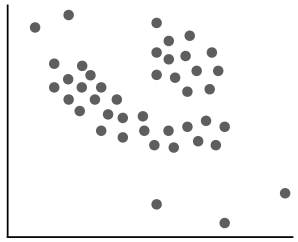


The Algorithm



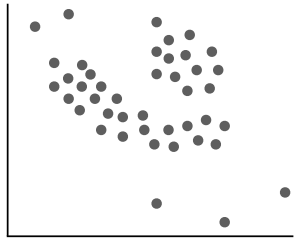
Hyperparameters

1. Distance metric
2. Epsilon (*eps*)
3. Minimum Points (*minpts*)



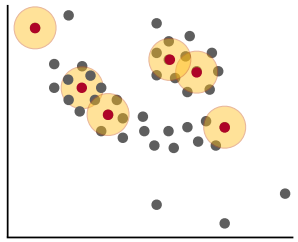
Hyperparameters

1. Distance metric
2. Epsilon (*eps*)
3. Minimum Points (*minpts*)



Hyperparameters

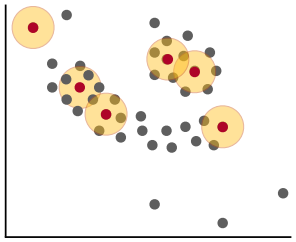
1. Distance metric
2. Epsilon (*eps*)
3. Minimum Points (*minpts*)



Hyperparameters

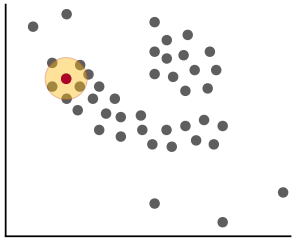
1. Distance metric
2. Epsilon (*eps*)
3. Minimum Points (*minpts*)

minimum number of points within *eps* distance of it in order to be considered dense



The Algorithm

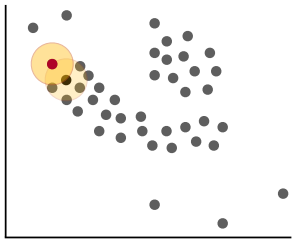
Core Point
A point is a core point if it has **at least** *minpts* neighbors within *eps* distance of itself.



The Algorithm

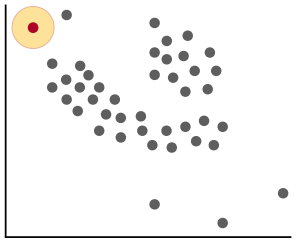
Core Point
A point is a core point if it has **at least** *minpts* neighbors within *eps* distance of itself.

Border Point
A point **without** at least *minpts* neighbors within *eps* distance of itself, **but** is a neighbor of a core point.



The Algorithm

these three types of points will define our clusters



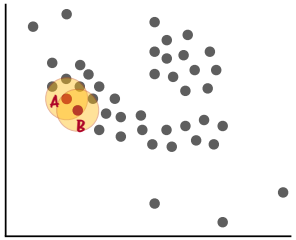
Core Point
A point is a core point if it has **at least** *minpts* neighbors within *eps* distance of itself.

Border Point
A point **without** at least *minpts* neighbors within *eps* distance of itself, **but** is a neighbor of a core point.

Noise
A point **without** at least *minpts* neighbors within *eps* distance of itself, **and is not a** neighbor of a core point.

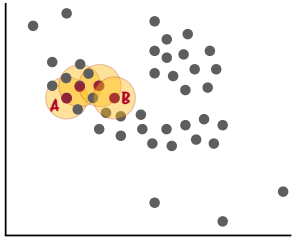
The Algorithm

Directly density reachable
Point A is directly density reachable from a core-point B if it is in the neighborhood of B.



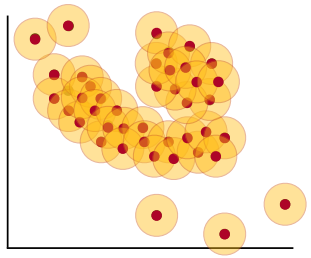
The Algorithm

Density reachable
Point A is density reachable from a core-point B if if there are a chain of points that are directly density reachable from B to A.



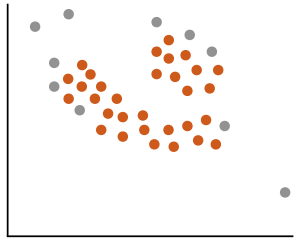
The Algorithm

1 Find all core points.



The Algorithm

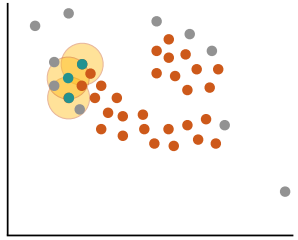
1 Find all core points.



The Algorithm

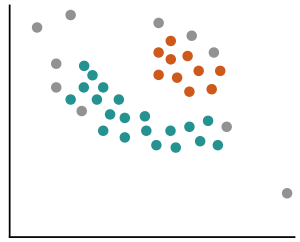
1 Find all core points.

2 Pick random core point, find other core points that are density reachable from it and assign to cluster.



The Algorithm

1. Find all core points.
2. Pick random core point, find other core points that are density reachable from it and assign to cluster.

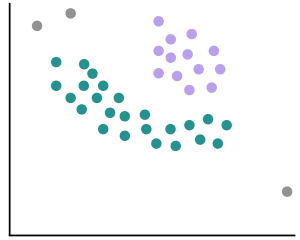


The Algorithm

1. Find all core points.
2. Pick random core point, find other core points that are density reachable from it and assign to cluster.
3. Add border points and cluster one is done.

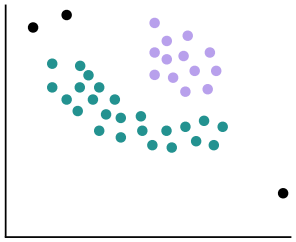
The Algorithm

1. Find all core points.
2. Pick random core point, find other core points that are density reachable from it and assign to cluster.
3. Add border points and cluster one is done.
4. Repeat from (2) until all clusters detected.

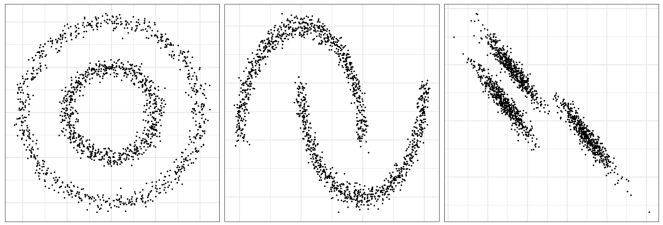


The Algorithm

1. Find all core points.
2. Pick random core point, find other core points that are density reachable from it and assign to cluster.
3. Add border points and cluster one is done.
4. Repeat from (2) until all clusters detected.
5. Remaining points are noise.

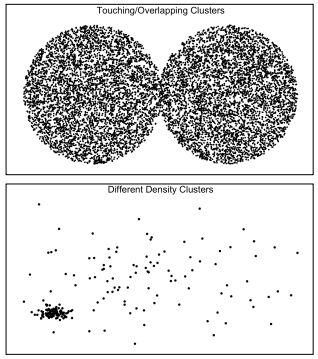


Example Data Structures



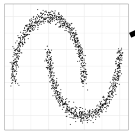
Where DBSCAN fails...

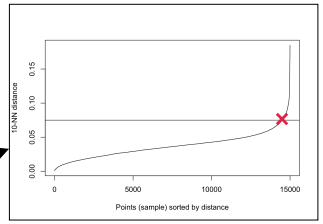
- Less effective on high dimensional data
- Overlapping/touching clusters
- Clusters have different densities



Hyperparameter Tuning

- Domain Knowledge + Distance Metrics
- More rows = larger *min_pts*
- More noise = larger *min_pts*
- More features = larger *min_pts*
- Elbow method: plot distances against data points to choose *eps*





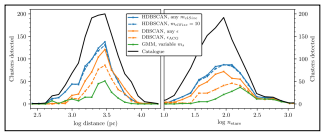
Applications

Figure 9 presents a sample image that was segmented using DBSCAN. In the figure, the individual clusters are regrouped together forming close to the original image. It can be observed that the pixels of similar color are clustered together.

ORIGINAL IMAGE SEGMENTATION USING DBSCAN

Figure 9: DBSCAN Example Result. [11]

[source]



[source]

This Week's Practical

Clustering