MLRF Lecture 01

J. Chazalon, LRE/EPITA, 2023

Clustering

Lecture 01 part 04

Clustering: finding groups in data (1/2)

Many techniques:

Useful for Practice session 1

- Connectivity models: hierarchical clustering, ...
  cluster = set of neighbors
- Centroid models: k-means, ...
  cluster = centroid point
- Distribution model: Gaussian mixtures models est. w. Expectation Maxim., …
  cluster = statistical distribution (center + spread in each direction)
- Density models: DBSCAN, …
  cluster = dense region
- Graph-based models: HCS (Highly Connected Subgraphs) algorithms, …
  cluster = clique in the graph
- ...

Useful for Practice session 4 (eventually)

Clustering: finding groups in data (2/2)

Always the same goal:

- Minimise the differences between elements within the same cluster
- Maximise the differences between elements within different clusters

Number of clusters:

- Many methods require to choose it beforehand
- Several techniques to adjust the number of clusters automatically

Outliers rejection:

- Some techniques do not assign lonely points to any cluster

⇒ Focus on HAC and K-Means today

Hierarchical Agglomerative Clustering (HAC)

Hierarchical Agglomerative Clustering

Algorithm

Initialization:

- Create a cluster from each element

While more than 1 cluster:

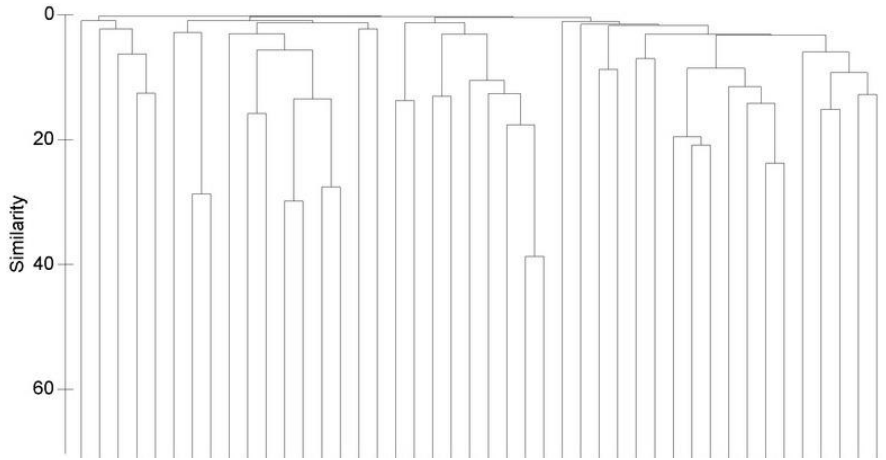
- Merge the two closest clusters

Challenge: distance between clusters

- New center = mean of points
- or distance = maximal distance
- or…

Time complexity: $O(n^2)$ for fastest method

Output: Dendrogram



Some linkage types

Single linkage minimizes the distance between the closest observations of pairs of clusters.

Maximum or complete linkage minimizes the maximum distance between observations of pairs of clusters.

Average linkage minimizes the average of the distances between all observations of pairs of clusters.

Centroid linkage first computes the centroid of clusters then looks at the distance between them.

Ward criterion minimizes the sum of squared differences within all clusters. (Much like K-Means.)

A word about Divisive clustering

HAC is *bottom-up*, divisive clustering is performed *top-down*.

Classical approach:

1. Start with all data
2. Apply flat clustering
3. Recursively apply the approach on each cluster until some termination

Pros: can have more than 2 sub-trees, **interesting for some indexing** (more on that in Lecture 4 with hierarchical K-Means), much faster than HAC

Cons: same issues as flat clustering, non-determinism

K-Means

K-Mean clustering (again) (1/3)

The K-means algorithm aims to choose centroids that minimise the inertia, or within-cluster sum-of-squares criterion:

$$\sum_{i=0}^{n} \min_{\mu_j \in C} (||x_i - \mu_j||^2)$$

- ie it does not maximizes inter-cluster distance
- ie it puts centers so as to get the best coverage (may not be on a density peak!)

K-Mean clustering (again) (2/3)

Algorithm

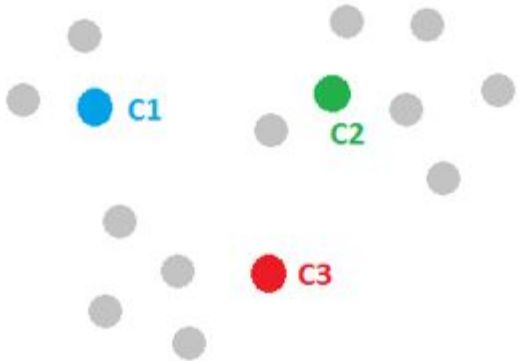
Images: https://healthcare.ai/step-step-k-means-clustering/

K-Mean clustering (again) (2/3)

Algorithm

Initialization:

- (randomly) select cluster centers



Images: https://healthcare.ai/step-step-k-means-clustering/

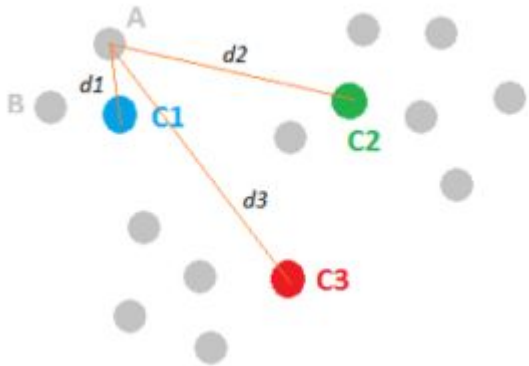
K-Mean clustering (again) (2/3)

Algorithm

Initialization:

- (randomly) select cluster centers

- Calculate distance points ⇔ centers



Images: https://healthcare.ai/step-step-k-means-clustering/

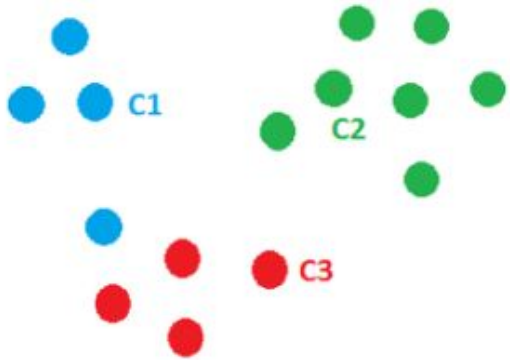
K-Mean clustering (again) (2/3)

Algorithm

Initialization:

- (randomly) select cluster centers

- Calculate distance points ⇔ centers
- Assign each point to closest center



Images: https://healthcare.ai/step-step-k-means-clustering/

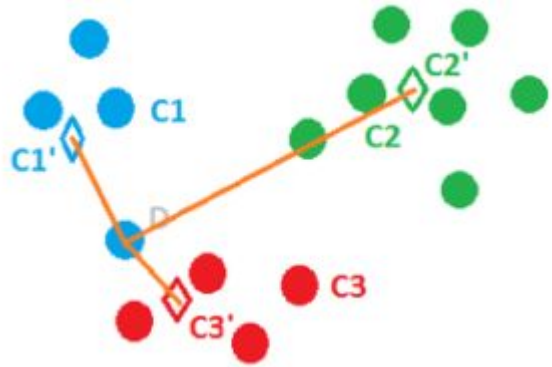
K-Mean clustering (again) (2/3)

Algorithm

Initialization:

- (randomly) select cluster centers

- Calculate distance points ⇔ centers
- Assign each point to closest center
- Update cluster centers: avg of points



Images: https://healthcare.ai/step-step-k-means-clustering/

K-Mean clustering (again) (2/3)

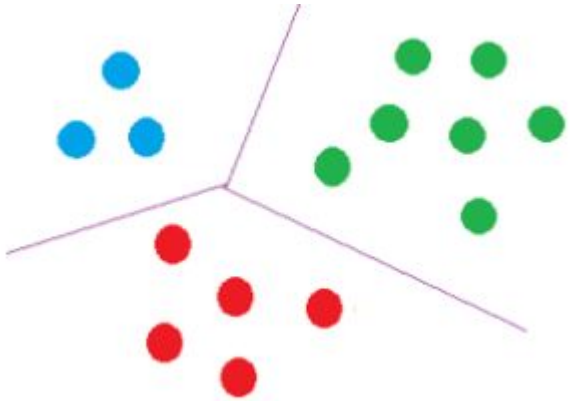
Algorithm

Initialization:

- (randomly) select cluster centers

<u>Loop until converged:</u>

- Calculate distance points ⇔ centers
- Assign each point to closest center
- Update cluster centers: avg of points



Result: centroid centers

- local maximas
- tessellation / Voronoi set over the dataset

Images: https://healthcare.ai/step-step-k-means-clustering/

K-Mean clustering (again) (3/3)

The previous algorithm is called **“Batch K-Means”**, or simply “**K-Means”**, because it considers the **whole the dataset at each iteration**.

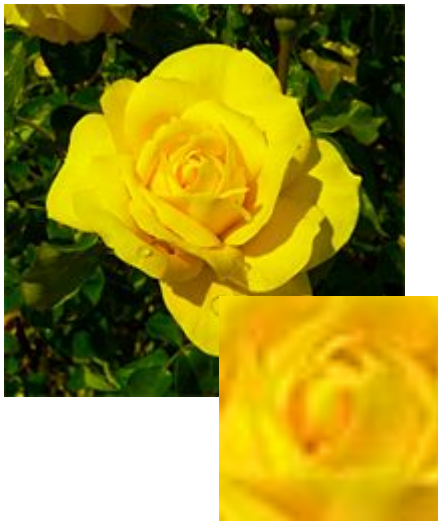
Batch K-Means is not only **sensible to outliers** and **initialization**, it is also **very slow** to compute on large datasets. (I got OOM errors with the *Twin it!* poster!!)

It is possible to avoid this speed / memory issue by **randomly sampling the dataset at each step.**

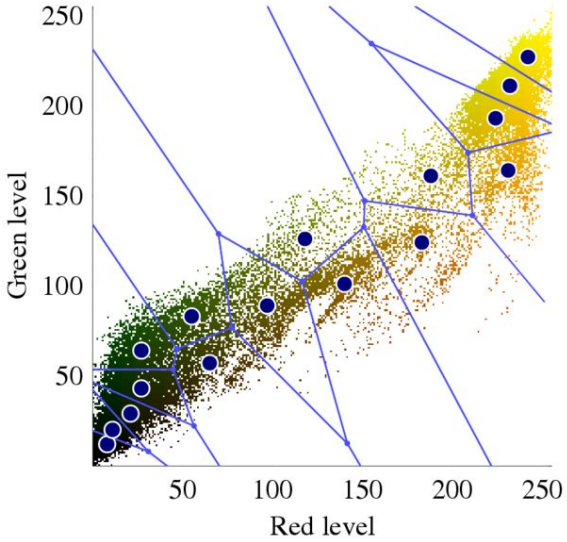
- Results are only slightly worse
- Speed and memory requirements make it usable on bigger datasets
- This approach is call **“Online K-Means”** or **“MiniBatch K-Means”**

Application: Color quantization

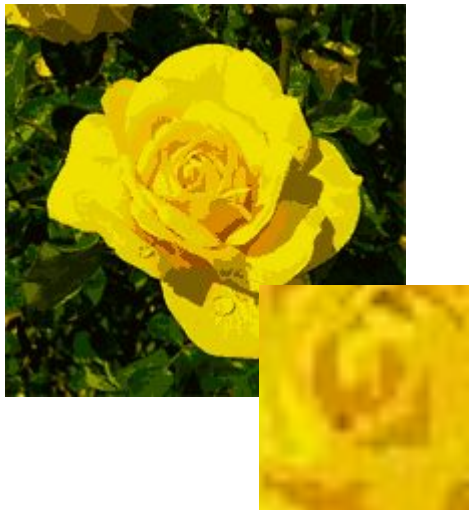
K=16



Original RG image
(no blue channel)



Color quantization showing original colors, target colors and boundaries (Voronoi cells here)

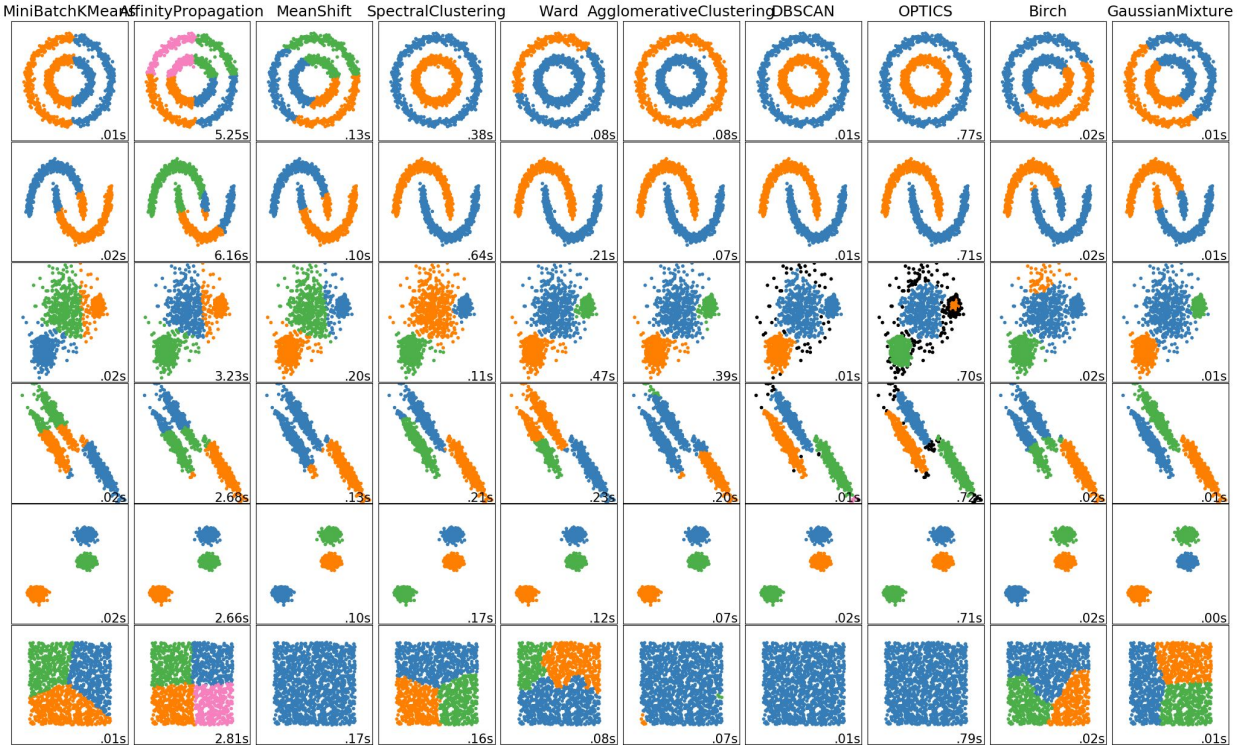


Color-indexed image
(no dithering)

Left/middle images by: Dcoetzee

Many clustering techniques to play with!

https://scikit-learn.org/stable/auto_examples/cluster/plot_cluster_comparison.html



Evaluation of Clustering

Need some supervision?

By construction, clustering algorithms are optimal as they are expect to find some optimal balance between high intra-cluster similarity and low inter-cluster similarity, on their training set.

How do these internal criteria translate into good effectiveness for applications?

A common approach is to rely on labeled data to compute new indicators:

- **Purity**: sort of “agreement” inside each cluster
- **Normalized Mutual Information** (NMI) and **Entropy**: information measures
- **Rand Index** (RI) and **F measure**: error counts

Check the IR-book for more details.

Modern density estimation point of view

But what about if we leave some samples out for testing the generalization?

HAC or K-Means “overtfit” the underlying data distribution.

It does not always make sense, but it we are interested in density estimation, then we can assess how well our model estimates the probability P(x) of unseen data. The “E” step of the EM algorithm is based on this idea.

Regularization could be used too, to reduce the variance of self-supervised density estimation techniques.