Introduction to Machine Learning

Lecture 4: Clustering

Alexis Zubiolo
alexis.zubiolo@gmail.com

Data Science Team Lead @ Adcash

November 17, 2016

What is clustering?

Clustering algorithms aim at **grouping unlabeled objects**: In the datasets, we have x, but no y!

What is clustering?

Clustering algorithms aim at **grouping unlabeled objects**: In the datasets, we have x, but no y!

Goal: Find clusters such that objects in the same cluster are more similar to each other than to objects in other clusters.

What is clustering?

Clustering algorithms aim at **grouping unlabeled objects**: In the datasets, we have x, but no y!

Goal: Find clusters such that objects in the same cluster are more similar to each other than to objects in other clusters.

Main **challenges**:

- What does *similar* mean?
- Given a similarity definition, how do we define clusters?
- How many clusters do we choose?

Clustering: A few applications

Ecology: Define comparisons of animal/plant communities over time

Web: Recognize communities of users

Marketing: Define groups of users with similar behavior/interests to target them more efficiently

Image processing: Segment images (*e.g.* segment different tissues in biomedical imaging)

And sometimes, we just have no labels in our data, so clustering can be a good first approach to tackle some problems.

Course outline

In this course, we will see 2 clustering algorithms:

Course outline

In this course, we will see 2 clustering algorithms:

- k-means
- Hierarchical clustering

k-means

k-means: Problem definition

Purpose: Split the set of points into k classes. k is a **parameter**!

k-means: Problem definition

Purpose: Split the set of points into k classes. k is a **parameter**!

We look for a partition $S = \{S_1, S_2, \ldots, S_k\}$ minimizing the within-cluster sum of squares.

$$\min_S \sum_{i=1}^{k} \sum_{x \in S_i} \|x - \mu_i\|_2^2$$

where

$$\mu_i = \frac{1}{|S_i|} \sum_{x \in S_i} x$$

is the centroid (mean) of points in S_i.

k-means: The algorithm

k-means applies the following 2 iterations:

k-means: The algorithm

k-means applies the following 2 iterations:

- Define the **Voronoi diagram** generated by the μ_is:

$$S_i^t = \left\{ x_p \mid \left\| x - \mu_i^t \right\| \leq \left\| x - \mu_j^t \right\|, 1 \leq j \leq k \right\}$$

k-means: The algorithm

k-means applies the following 2 iterations:

- Define the **Voronoi diagram** generated by the μ_is:

$$S_i^t = \left\{x_p \mid \left\|x - \mu_i^t\right\| \leq \left\|x - \mu_j^t\right\|, 1 \leq j \leq k\right\}$$

- Update the centroid:

$$\mu_i^{t+1} = \text{centroid}(S_i^t) = \frac{1}{S_i^t} \sum_{x \in S_i^t} x$$

for all $i \in \{1, \ldots, k\}$

k-means: The algorithm

k-means applies the following 2 iterations:

- Define the **Voronoi diagram** generated by the μ_is:

$$S_i^t = \left\{ x_p \mid \left\| x - \mu_i^t \right\| \leq \left\| x - \mu_j^t \right\|, 1 \leq j \leq k \right\}$$

- Update the centroid:

$$\mu_i^{t+1} = \mathrm{centroid}(S_i^t) = \frac{1}{S_i^t} \sum_{x \in S_i^t} x$$

for all $i \in \{1, \ldots, k\}$

Problem in the above formulas: **initial value** for the μ_is?

k-means: Initialization

Remark: The k-means solution depends on the initial position of the μ_is centroids.
(see animation by Andrey Shabalin)

k-means: Initialization

Remark: The k-means solution depends on the initial position of the μ_is centroids.
(see animation by Andrey Shabalin)

2 related questions:

1. How to choose the initial μ_is?
2. How to have more stable results?

k-means: Initialization

Remark: The k-means solution depends on the initial position of the μ_is centroids.
(see animation by Andrey Shabalin)

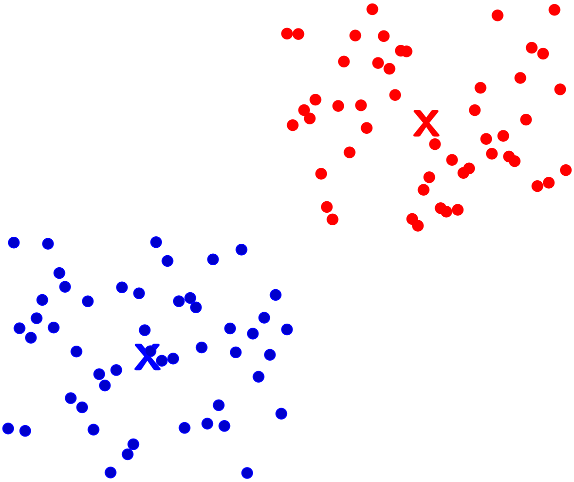
2 related questions:

1. How to choose the initial μ_is?
2. How to have more stable results?

Unfortunately, no miracle strategy for Q1. A common strategy:

- Several k-means with random initializations
- Majority vote

$k = 2$



Speeding up k-means

Each k-means iteration is done over all the points in the dataset. This can be computationally expensive, especially if

- There are many points
- The point density is big

What to do to **speed up the process**?

Speeding up k-means

Each k-means iteration is done over all the points in the dataset. This can be computationally expensive, especially if

- There are many points
- The point density is big

What to do to **speed up the process**?

Alternative: **Mini-batch** k-means. At each iteration

- Choose a subset of points
- Apply a k-means iteration

Number of clusters

In some applications, you know how many clusters you want. In this case, k is **easy to set**.

In other applications, we don't know the optimal number of classes we want. Ideally, we would like k to be selected automatically.

There is always some ambiguity in selection the *optimal* number of clusters. This is normal: When doing unsupervised learning, there is necessarily some inherent subjectivity in the labeling process!

Number of clusters

That being said, it is possible to define some criterias to determine whether k_1 is a better number of clusters than k_2. We can use the sum of squared errors to the centroids:

$$\mathrm{SSE}(k) = \sum_{i=1}^{k} \sum_{x \in S_i} \|x - \mu_i\|_2^2$$

And apply the **Elbow method**.

Number of clusters

That being said, it is possible to define some criterias to determine whether k_1 is a better number of clusters than k_2. We can use the sum of squared errors to the centroids:

$$\mathrm{SSE}(k) = \sum_{i=1}^{k} \sum_{x \in S_i} \|x - \mu_i\|_2^2$$

And apply the **Elbow method**.

Note that this is not a miracle solution.

k-means: concluding remarks

Pros:

-
-

k-means: concluding remarks

Pros:

-
-

Cons:

-

k-means: concluding remarks

Pros:

-
-

Cons:

-

We can use different metrics to link clusters and various linkage criteria.

Hierarchical clustering

Hierarchical clustering

Bottom-up approach:

- **Bottom**: Initially, each point is a cluster
- **Top**: Merge the 2 closest clusters until we have one cluster

Hierarchical clustering

Bottom-up approach:

- **Bottom**: Initially, each point is a cluster
- **Top**: Merge the 2 closest clusters until we have one cluster

closest has to be defined. We can use the **Euclidean norm**: a and b are closer than a' and b' if and only if $\|a - b\| \leq \|a' - b'\|$.

Hierarchical clustering

Bottom-up approach:

- **Bottom**: Initially, each point is a cluster
- **Top**: Merge the 2 closest clusters until we have one cluster

closest has to be defined. We can use the **Euclidean norm**: a and b are closer than a' and b' if and only if $\|a - b\| \leq \|a' - b'\|$.

(illustration on the clipboard)

Hierarchical clustering: Dendograms

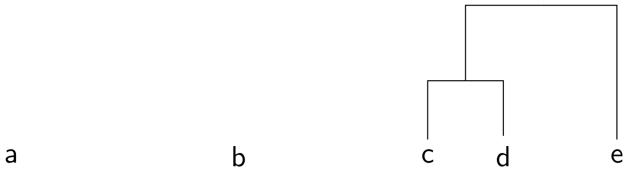
a b c d e

Formally, this can be done with a **linkage criterion**.

Hierarchical clustering: Dendograms

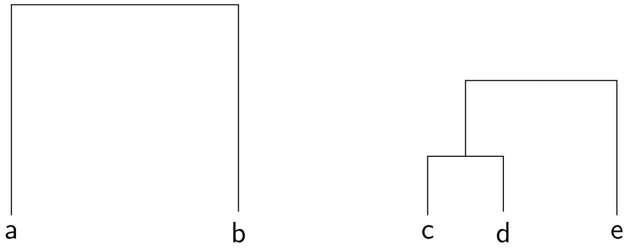
Formally, this can be done with a **linkage criterion**.

Hierarchical clustering: Dendograms



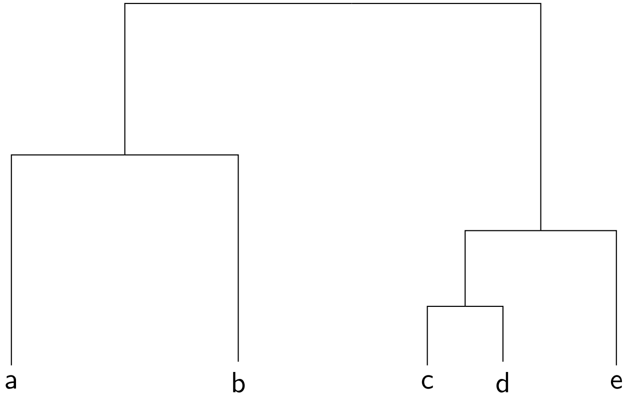
Formally, this can be done with a **linkage criterion**.

Hierarchical clustering: Dendograms



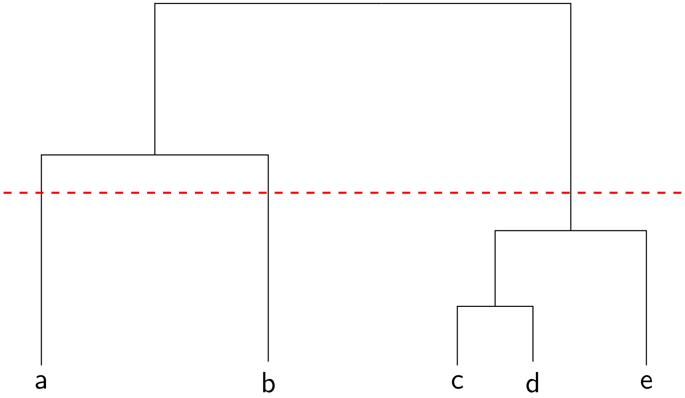
Formally, this can be done with a **linkage criterion**.

Hierarchical clustering: Dendograms



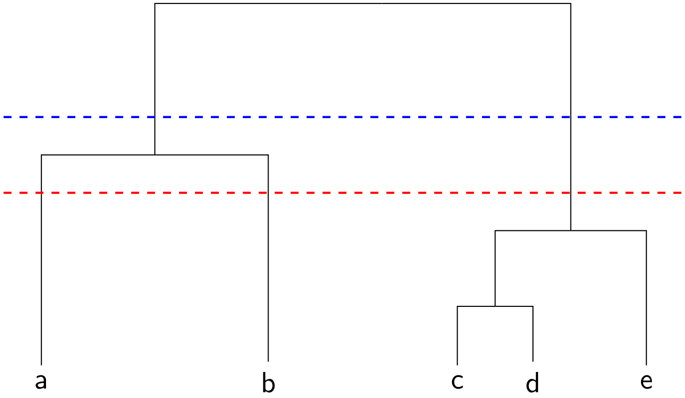
Formally, this can be done with a **linkage criterion**.

Hierarchical clustering: Dendograms



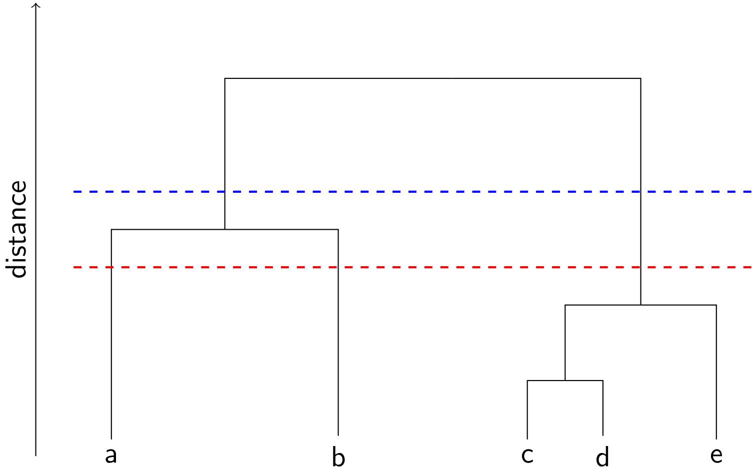
Formally, this can be done with a **linkage criterion**.

Hierarchical clustering: Dendograms



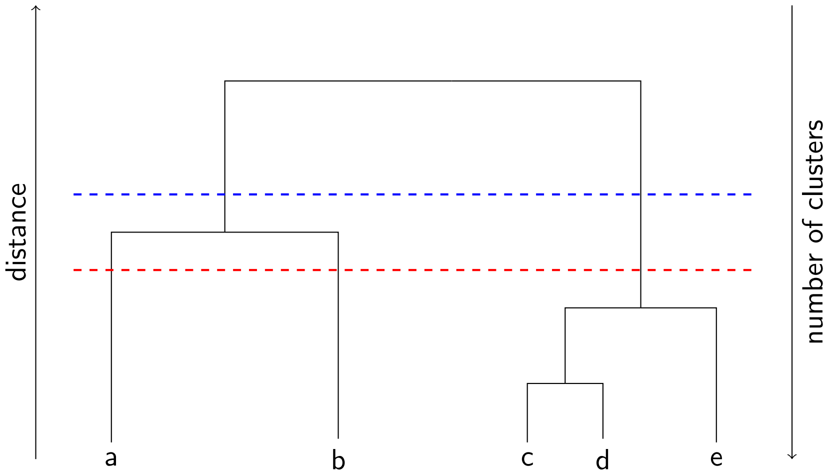
Formally, this can be done with a **linkage criterion**.

Hierarchical clustering: Dendograms

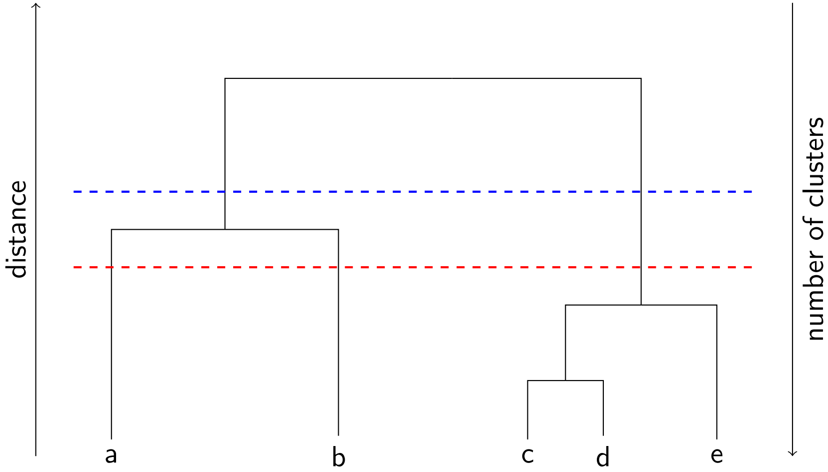


Formally, this can be done with a **linkage criterion**.

Hierarchical clustering: Dendograms



Hierarchical clustering: Dendograms



Formally, this can be done with a **linkage criterion**.

Hierarchical clustering: concluding remarks

Pros:

- No need to provide an initial number of classes
- Can split the clusters at different levels

Hierarchical clustering: concluding remarks

Pros:

- No need to provide an initial number of classes
- Can split the clusters at different levels

Cons:

- Similarity matrix: need to compute the distances between all pairs of points in the dataset

Hierarchical clustering: concluding remarks

Pros:

- No need to provide an initial number of classes
- Can split the clusters at different levels

Cons:

- Similarity matrix: need to compute the distances between all pairs of points in the dataset

We can use different metrics to link clusters and various linkage criteria.

Hierarchical clustering: concluding remarks

Pros:

- No need to provide an initial number of classes
- Can split the clusters at different levels

Cons:

- Similarity matrix: need to compute the distances between all pairs of points in the dataset

We can use different metrics to link clusters and various linkage criteria.

Top-down approach also possible.

Conclusion

Clustering methods create groups of unlabeled points.

Conclusion

Clustering methods create groups of unlabeled points.

They typically rely on:

- A similarity (or dissimilarity) measure (*e.g.* the Euclidian norm)
- A few parameters, *e.g.*
  - The number of clusters for k-means
  - The dendogram cut (linkage) for hierarchical clustering

Thank you! Questions?