Computational Tools for Data Science

Week 6:

Clustering

Motivation: Clustering

- **What is it:** Task of **grouping** objects s.t. objects of the same group are more "similar" to each other than objects from different ones. The groups are called **"clusters".** In general they may overlap.

- **For which data**: numerical, usually a lot of data and high-dimensional

- **Usually:**

  – Objects are points (or vectors) in a high-dimensional (vector) space.

  – "Similarity" is defined via some distance measure, e.g.:

    » vectors: Euclidean distance, Cosine

    » Jaccard similarity (for sets), edit distance (strings)

Reminder: Distance

- **Definition (distance d):** Let V be a (top.) space. A map $d: V \times V \to \mathbb{R}$ is called a *distance* if the following properties hold:

  1) For all $u, v \in V$: $d(u,v) \geq 0$ and $d(u,v) = 0$ iff $u = v$.

  2) For all $u, v \in V$: $d(u,v) = d(v,u)$

  3) For all $\mathrm{u}, v, w \in V$: $d(u,v) + d(v,w) \geq d(u,w)$. [Triangular inequality]

Motivation: Clustering
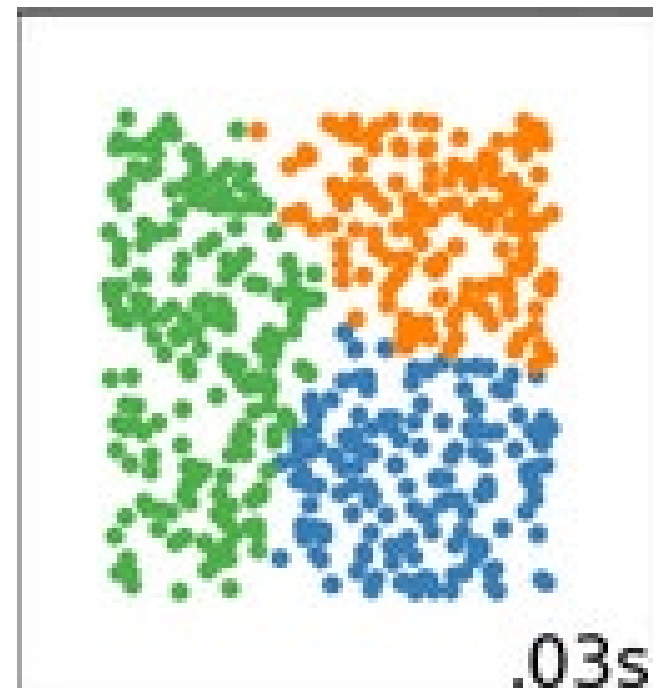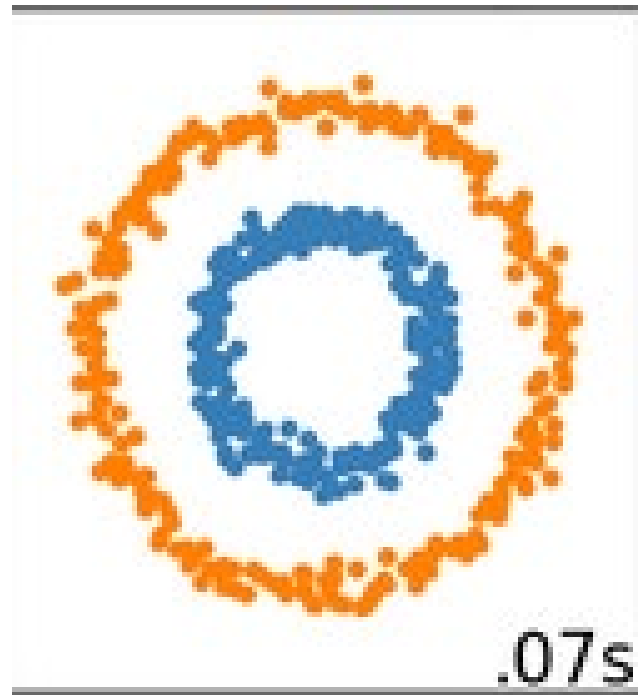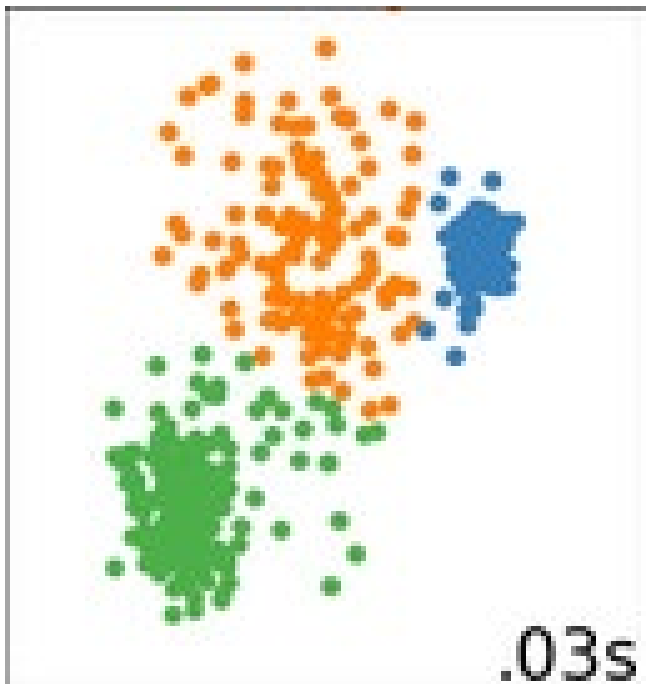
Motivation: Clustering

Examples:

- In general: categorisation, classification and segmentation
    - » But also: finding anomalies, recognition

- Clustering stelar objects (clusters of stars)

- Group people by interest in social media (via #s, followers, topics)

- Group people / customer by consumed goods (e.g. music, movies, food)

- Find categories for goods by customers who bought them

- Identify topics for documents via shared words

Motivation: Clustering

Example: Categorise goods, say **movies**, via some set C of customers

- Represent each movie as $|C|$ dimensional 0-1 vector, one dimension per customer from C.

- An entry / component in this vector is 1 iff the corresponding customer has watched / purchased the movie.

- $\rightarrow$ This yields thousands or even millions of dimensions (e.g. Amazon)

- Idea behind: Taste in movies causes us to buy / watch similar movies.

Hierarchical Clustering

A) Agglomerative (bottom up):

1. Initialise each data point as a (singleton) cluster.

2. Repeatedly combine two "most similar / closest" clusters to a new one.

3. Stop w.r.t. certain criterion (e.g. # of clusters, density of clusters, …)
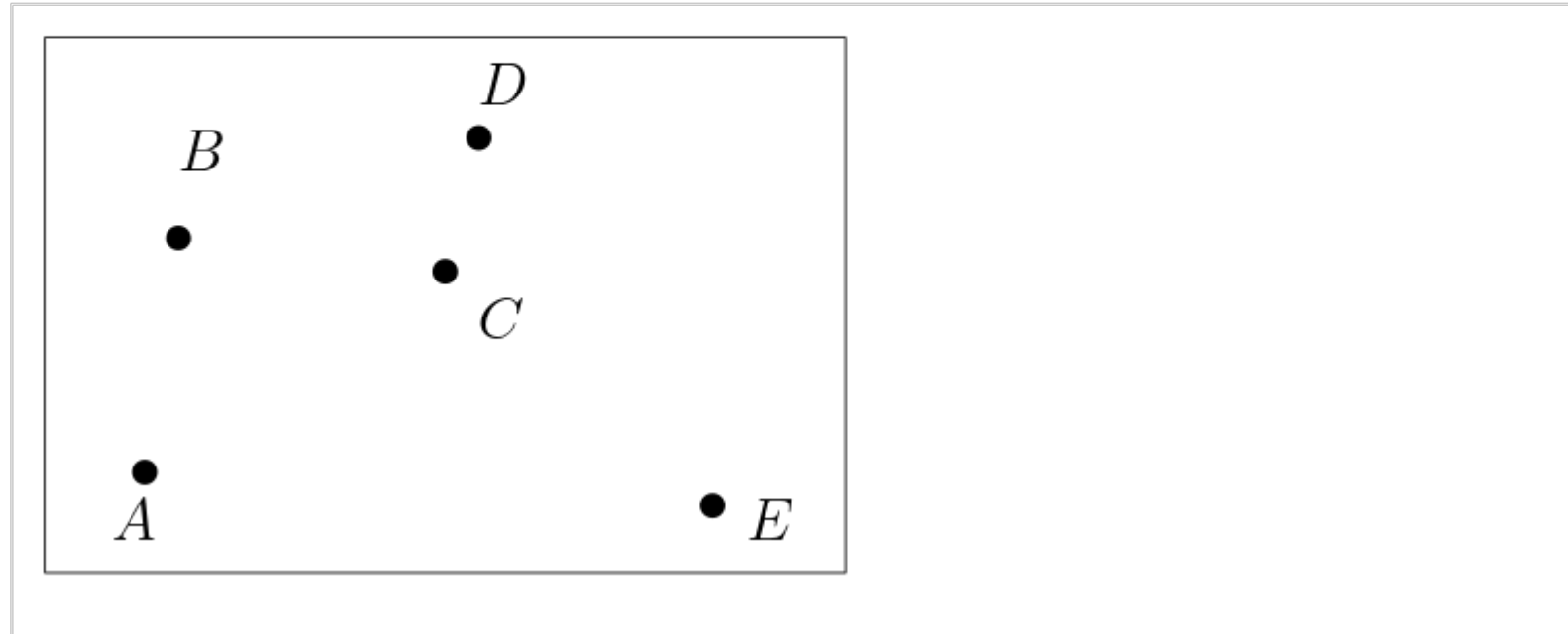
Hierarchical Clustering

A) Agglomerative (bottom up):

1. Initialise each data point as a (singleton) cluster.
2. Repeatedly combine two "most similar / closest" clusters to a new one.
3. Stop w.r.t. certain criterion (e.g. # of clusters, density of clusters, …)

B) Divisive (top down):

1. Initialise the whole data set as one big cluster.
2. Recursively split up cluster which is least "dense / connected".
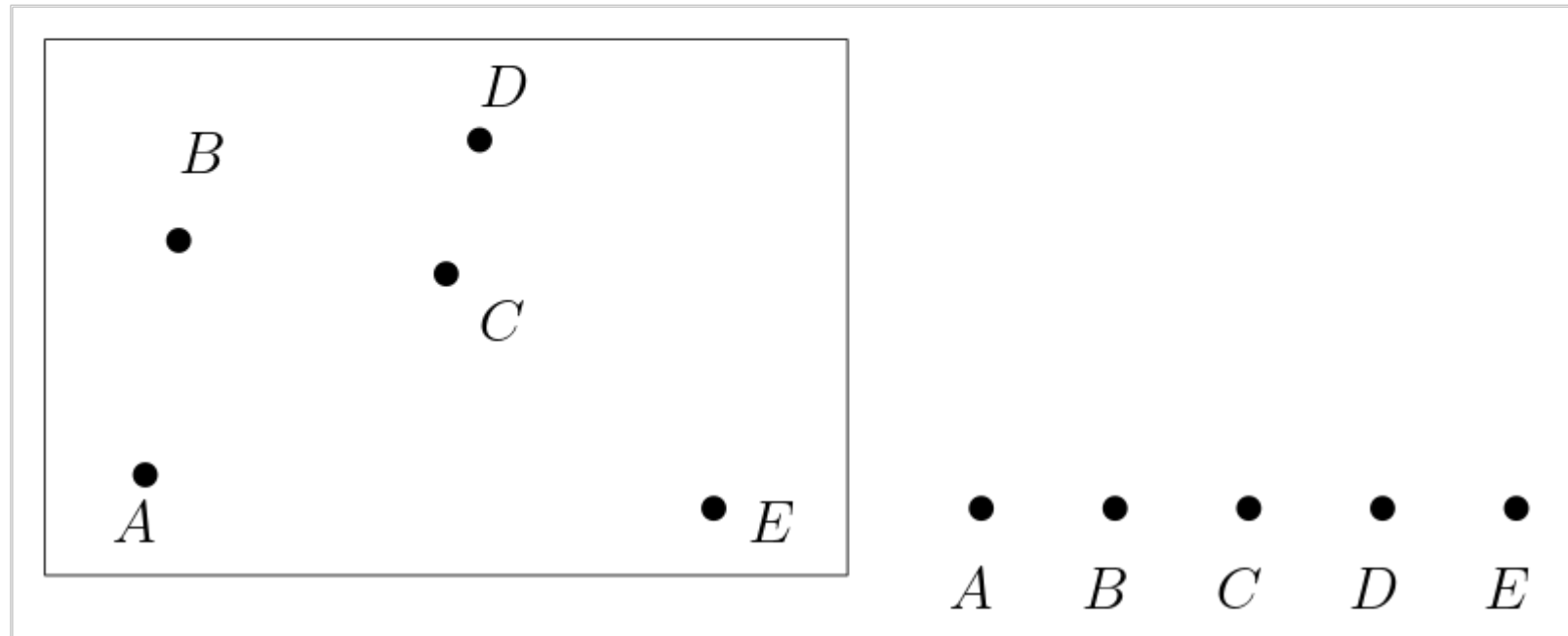3. Stop w.r.t. certain criterion (e.g. # of clusters, density of clusters, …)

Small example

Resulting **tree structure** called *dendrogram*.

Use case: E.g. in **phylogenetics** with no huge data (**phylogenetic tree**)

Small example

Resulting **tree structure** called *dendrogram*.

Use case: E.g. in **phylogenetics** with no huge data (**phylogenetic tree**)
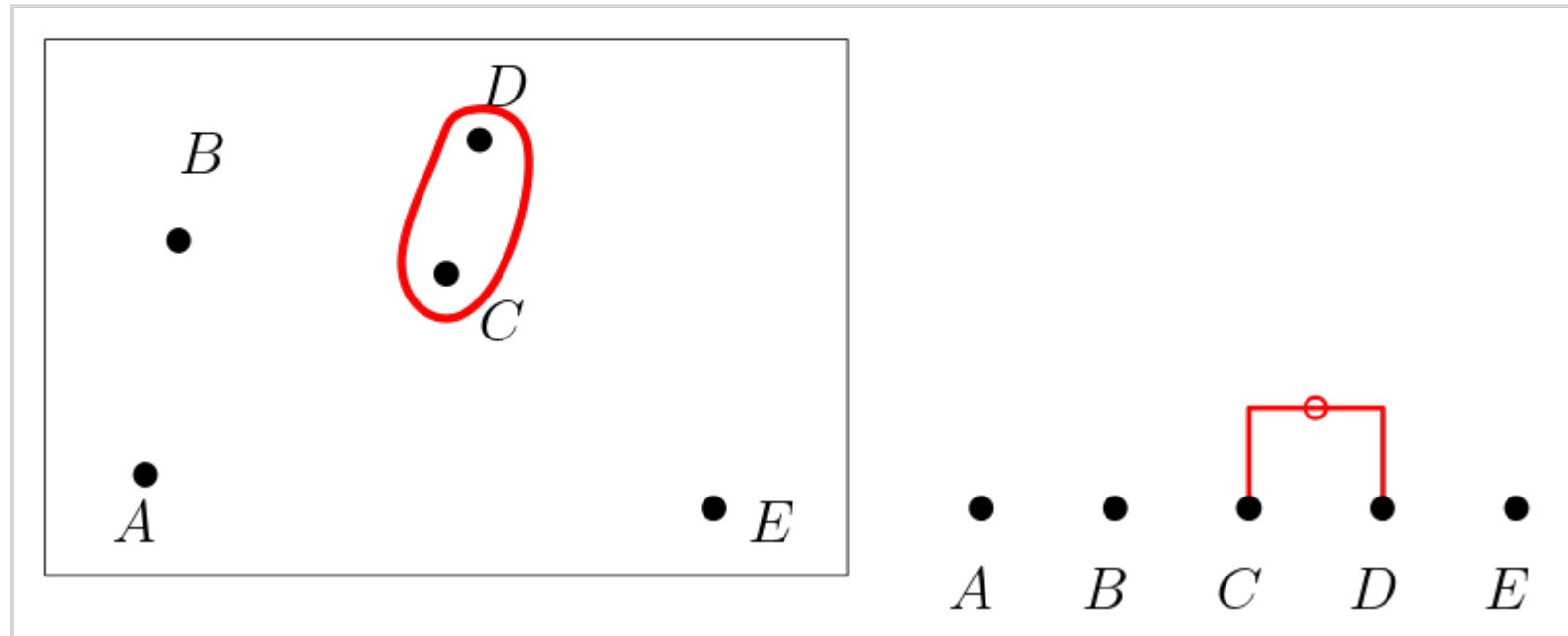
Small example

Resulting **tree structure** called *dendrogram*.

Use case: E.g. in **phylogenetics** with no huge data (**phylogenetic tree**)

Small example

Resulting **tree structure** called *dendrogram*.

Use case: E.g. in **phylogenetics** with no huge data (**phylogenetic tree**)

Small example

Resulting **tree structure** called *dendrogram*.

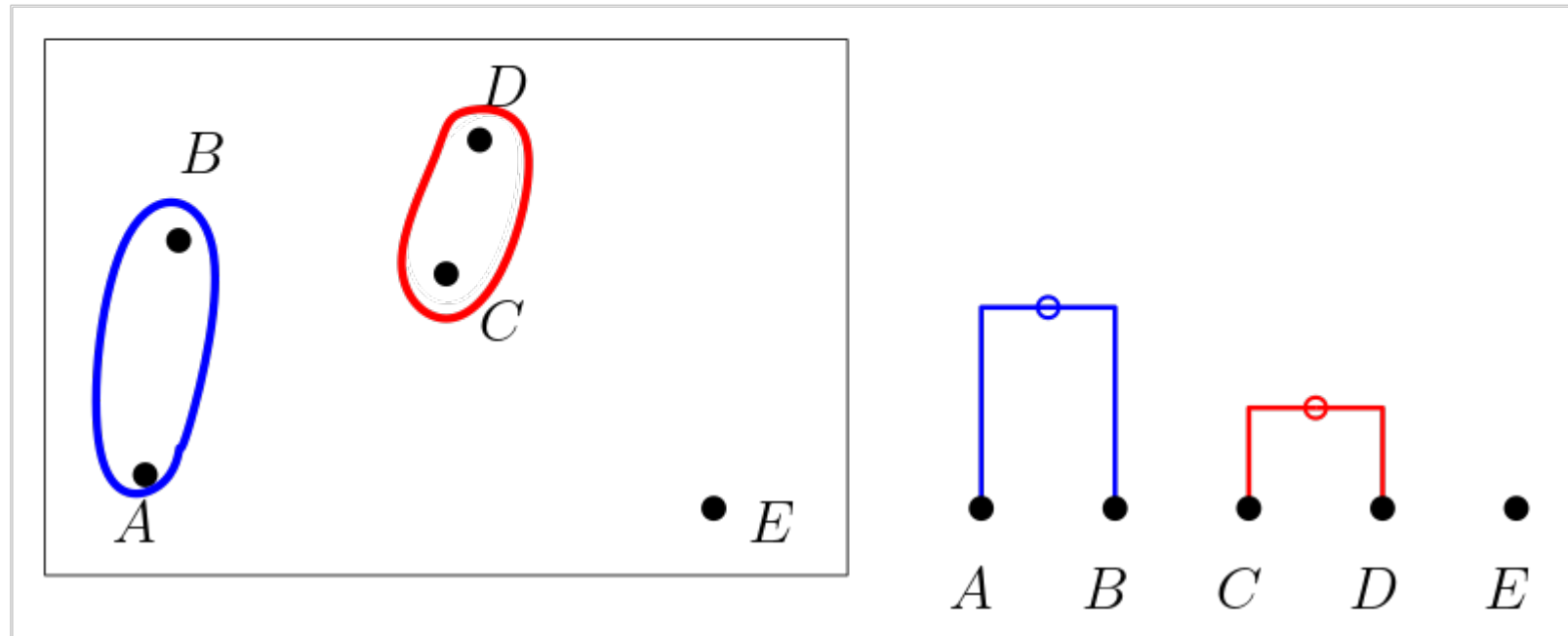**Use case:** E.g. in **phylogenetics** with no huge data (**phylogenetic tree**)
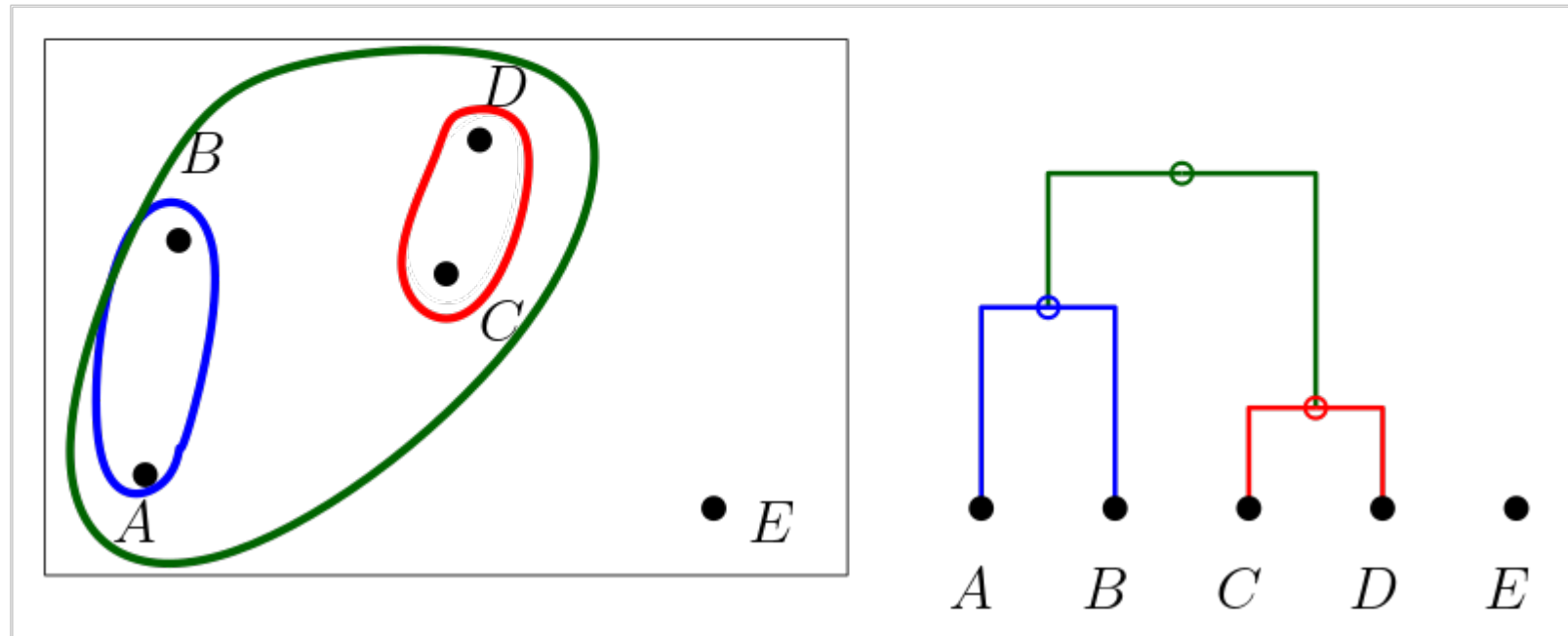
Small example

Resulting **tree structure** called *dendrogram*.

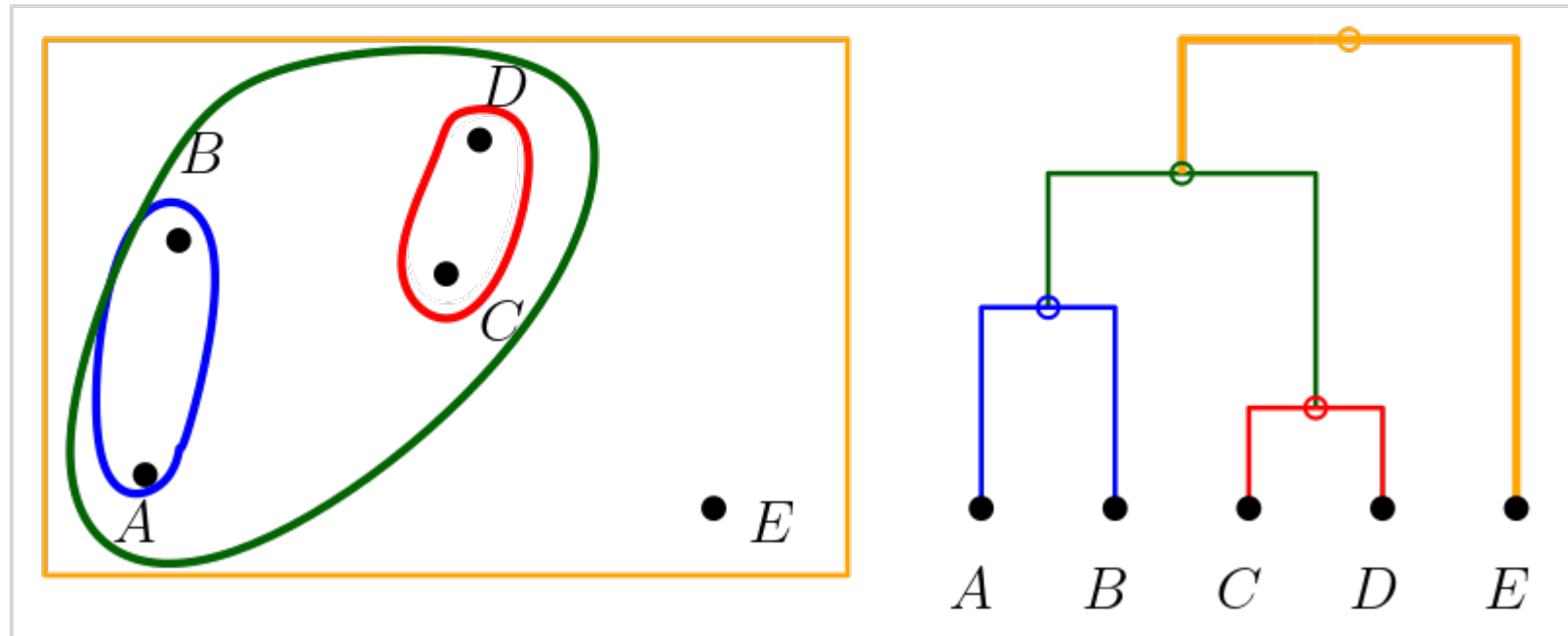**Use case:** E.g. in **phylogenetics** with no huge data (**phylogenetic tree**)

(Hierarchical) Clustering

3 important questions to answer:

a) How do we represent clusters?

b) How do we determine "similarity / closeness" of clusters?

c) When do we stop the algorithm?

Representation of clusters

Euclidean case:

Let C be a cluster of n many d-dimensional data points, say in $\mathbb{R}^d$. We define the **centroid** $cent(C)$ of C as follows:

$$cent(C) = \frac{1}{n} \sum_{c \in C} c$$

Think of **centroid** as a **centre of gravity**.

Representation of clusters

Non-Euclidean case:

Let C be a cluster of n many data points.

Define the **clustroid** $clust(C)$ of C as a data point (from C) that is "<u>closest</u>" to the other points in C.

Some possible meanings of "<u>closest</u>":

Representation of clusters

Non-Euclidean case:

Let C be a cluster of n many data points.

Define the **clustroid** $clust(C)$ of C as a data point (from C) that is "<u>closest</u>" to the other points in C.

Some possible meanings of "<u>closest</u>":

• Smallest maximum distance to other points in C. → $\min\limits_{c \in C} \max\limits_{v \in C} d(c, v)$

Representation of clusters

Non-Euclidean case:

Let C be a cluster of n many data points.

Define the **clustroid** $clust(C)$ of C as a data point (from C) that is "<u>closest</u>" to the other points in C.

Some possible meanings of "<u>closest</u>":

- Smallest maximum distance to other points in C. $\rightarrow \min_{c \in C} \max_{v \in C} d(c, v)$

- Smallest average distance to other points in C. $\rightarrow \min_{c \in C} \frac{1}{n} \sum_{v \in C} d(c, v)$

Representation of clusters

Non-Euclidean case:

Let C be a cluster of n many data points.

Define the **clustroid** $clust(C)$ of C as a data point (from C) that is "<u>closest</u>" to the other points in C.

Some possible meanings of "<u>closest</u>":

• Smallest maximum distance to other points in C. → $\min\limits_{c \in C} \max\limits_{v \in C} d(c, v)$

• Smallest average distance to other points in C. → $\min\limits_{c \in C} \frac{1}{n} \sum_{v \in C} d(c, v)$

• Smallest sum of squared distances to other points in C.

$$\rightarrow \min\limits_{c \in C} \sum_{v \in C} d^2(c, v)$$

Similarity / closeness of clusters

Idea: Replace each cluster by its centroid (Euclidean) or clustroid (Non-Euclidean).

Measure distance of two clusters C and C' to each other by:

Euclidean setting: $\qquad d(cent(C), cent(C'))$

Non-Euclidean setting: $\quad d(clust(C), clust(C'))$

Similarity / closeness of clusters

Other ways to measure:

- Define distance between two clusters C and C' as the minimum distance between **any** data **point from** C and **any** data **point in** C'.

- Set a notion of "**cohesion**" (or connectivity, density, …)

  $\rightarrow$ Merge two cluster whose union is most cohesive.

Similarity / closeness of clusters

Set a notion of "**cohesion**" (or connectivity, density, …)

$\rightarrow$ Merge two cluster whose union is most cohesive.

Possible notions of "cohesion":

A) Minimum **diameter**

For a cluster C define its diameter $diam(C)$ as:

$$diam(C) = \max_{c,c' \in C} d(c, c')$$

Similarity / closeness of clusters

Set a notion of "**cohesion**" (or connectivity, density, …)

→ Merge two cluster whose union is most cohesive.

Possible notions of "cohesion":

B) Minimum **average distance** (between two points in the cluster)

Let C be a cluster with $n := |C|$ many points in it. Further let $\binom{C}{2}$ denote the set of all pairs of points in C. Then the average distance $avd(C)$ of C is:

$$avd(C) := \frac{2}{n(n-1)} \sum_{\{c,c'\} \in \binom{C}{2}} d(c,c')$$

Similarity / closeness of clusters

Set a notion of "**cohesion**" (or connectivity, density, …)

$\rightarrow$ Merge two cluster whose union is most cohesive.

Possible notions of "cohesion":

C) Minimum **density**

E.g. define the density of a cluster C by: $\frac{diam(C)}{|C|}$ or $\frac{avd(C)}{|C|}$

Evaluating the clustering (Davies-Bouldin index)

Definition (Davies-Bouldin index):

Let $k \in \mathbb{N}$ and $C_1, \dots, C_k$ be the clusters of some data set.

Let $c_i := cent(C_i)$ for all $i \in \{1, \dots, k\}$.

Define the ***average radius*** r_i of a cluster C_i as:

$$r_i := \frac{1}{|C_i|} \sum_{c \in C_i} d(c_i, c)$$

Define the **Davies-Bouldin index** $DB(C_1, \dots, C_k)$ for $C_1, \dots, C_k$ as:

$$DB(C_1, \dots, C_k) := \frac{1}{k} \sum_{i=1}^{k} \max_{j \neq i} \frac{r_i + r_j}{d(c_i, c_j)}$$

Note: Index is low if average radii are small and clusters are far apart.

(Naive) Hierarchical clustering algorithm

Agglomerative (bottom up):

1. Initialise each data point as a (singleton) cluster.
    AND initialise representatives for clusters (centroid / clustroid).

2. Repeatedly combine two "most similar / closest" clusters to a new one.
    BY computing in each turn **distances** between **all pairs** of clusters.

3. Stop w.r.t. certain criterion (e.g. # of clusters, density of clusters, …)
    BY computing abort parameter each turn.

(Naive) Hierarchical clustering algorithm

2. Repeatedly combine two "most similar / closest" clusters to a new one.

   BY computing in each turn **distances** between **all pairs** of clusters.

If we have n data points in total. This part of the algorithm takes $O(n^3)$ much time.

Conclusion: Hierarchical clustering takes too long for large data sets!

The (naive) k-means algorithm

- Sometimes also called **Lloyd's algorithm**, named after Stuart P. Lloyd.

- During the algorithm, the **number of clusters** is **fixed** and equal to ***k***.

- **Point assignment method**, no hierarchical approach.
    - Every data point will be assigned to precisely one cluster.

- **Iterative algorithm** that assumes **Euclidean setting**.

- The algorithm alternates between two steps:

    - **Assignment step**

    - **Update step**

The (naive) k-means algorithm

Step 0: Initialisation

- Choose k data points as representatives (centroids), one for each (singleton) cluster $C_1, \ldots, C_k$.

  - e.g. all at random, or

  - one at random and each next as far away from former(s) as possible

Assignment step:

(Re)assign each data point p to the cluster C_i for which the distance between *p* and the centroid of C_i is minimal,

i.e. to the cluster C_i that minimises: $d(p, cent(C_i))$.

The (naive) k-means algorithm

Assignment step:

(Re)assign each data point p to the cluster C_i for which the distance between p and the centroid of C_i is minimal,

i.e. to the cluster C_i that minimises: $d(p, cent(C_i))$.

Update step:

Recalculate the centroids for all clusters.

Termination rule:

– E.g. run until the process converges / stabilises, or

– stop if only a few / tiny proportion of data points were reassigned.

(Naive) k-means algorithm: running time

Let $k \in \mathbb{N}$ be fixed

Let n be the **total number of data points** we want to cluster.

Let I denote the **number of iteration** the k-means algorithm needs **until convergence** / it stabilises (abort criterion: no changes anymore).

Then the **running time** is: $O(knI)$.

How big is I?

Worst case: $I = 2^{\Omega(\sqrt{n})}$

In practice: I can often be considered **constant**.

The k-means Algorithm
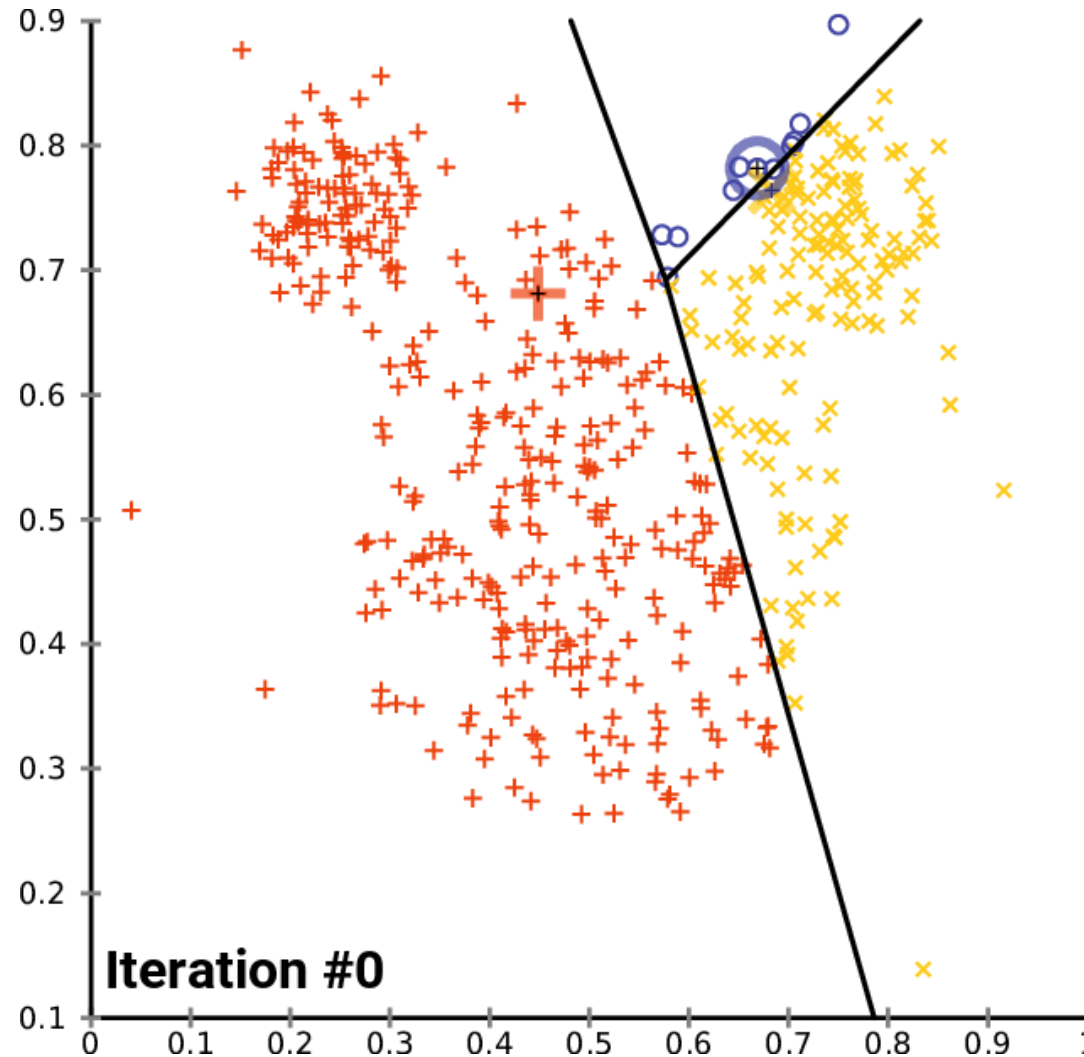
GIF published on Wikipedia under CC BY-SA 4.0 license by Chire. See this link.

What might the right *k* be?

Let $k \in \mathbb{N}$ and $C_1, \ldots, C_k$ be the clusters of some data set.

Let $c_i := cent(C_i)$ for all $i \in \{1, \ldots, k\}$.

Define the **average radius** r_i of a cluster C_i as:

$$r_i := \frac{1}{|C_i|} \sum_{c \in C_i} d(c_i, c)$$

Often: Average radii get rapidly smaller while we increase k, up to some value from where on changes are relatively small.

Compute evaluation score: (e.g. Davies-Bouldin index)

Problem with k-means

In general, k-means (tries) to minimise the following term:

$$\sum_{i=1}^{k} \sum_{c \in C_i} d(c, cent(C_i))$$

Problematic, if (actual) clusters differ a lot in size or are not well-distributed.
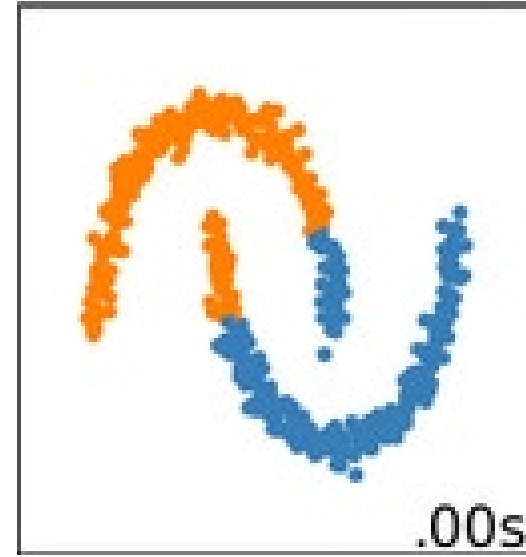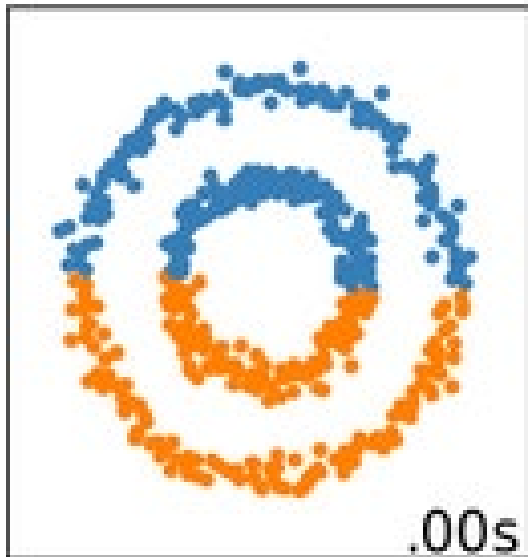
CURE (Clustering Using REpresentatives) algorithm

An algorithm "between" agglomerative and k-means clustering.

Step 0: Pick **random sample** of data points fitting into main memory.

Step 1: Cluster the sample set, e.g. via agglomerative clustering.

Step 2:

- Within each cluster C_i, pick a sample set S_i of data points which is as "***dispersed***" as possible.
- Define **set of representatives** $\boldsymbol{R_i}$ of C_i as those points we get by moving each point of S_i by some fraction towards $cent(C_i)$.

Note: Points in R_i are not necessarily actual data points.

CURE (Clustering Using REpresentatives) algorithm

Next: Merge two clusters if there are two representatives, one from each cluster, whose distance from each other is below some fixed threshold.

$\rightarrow$ pick new scattered representatives

Now start clustering of the whole data by the following rule:

Assignment rule: Place data point p into cluster C_i if p is closest to a representative of C_i among representatives of all clusters.
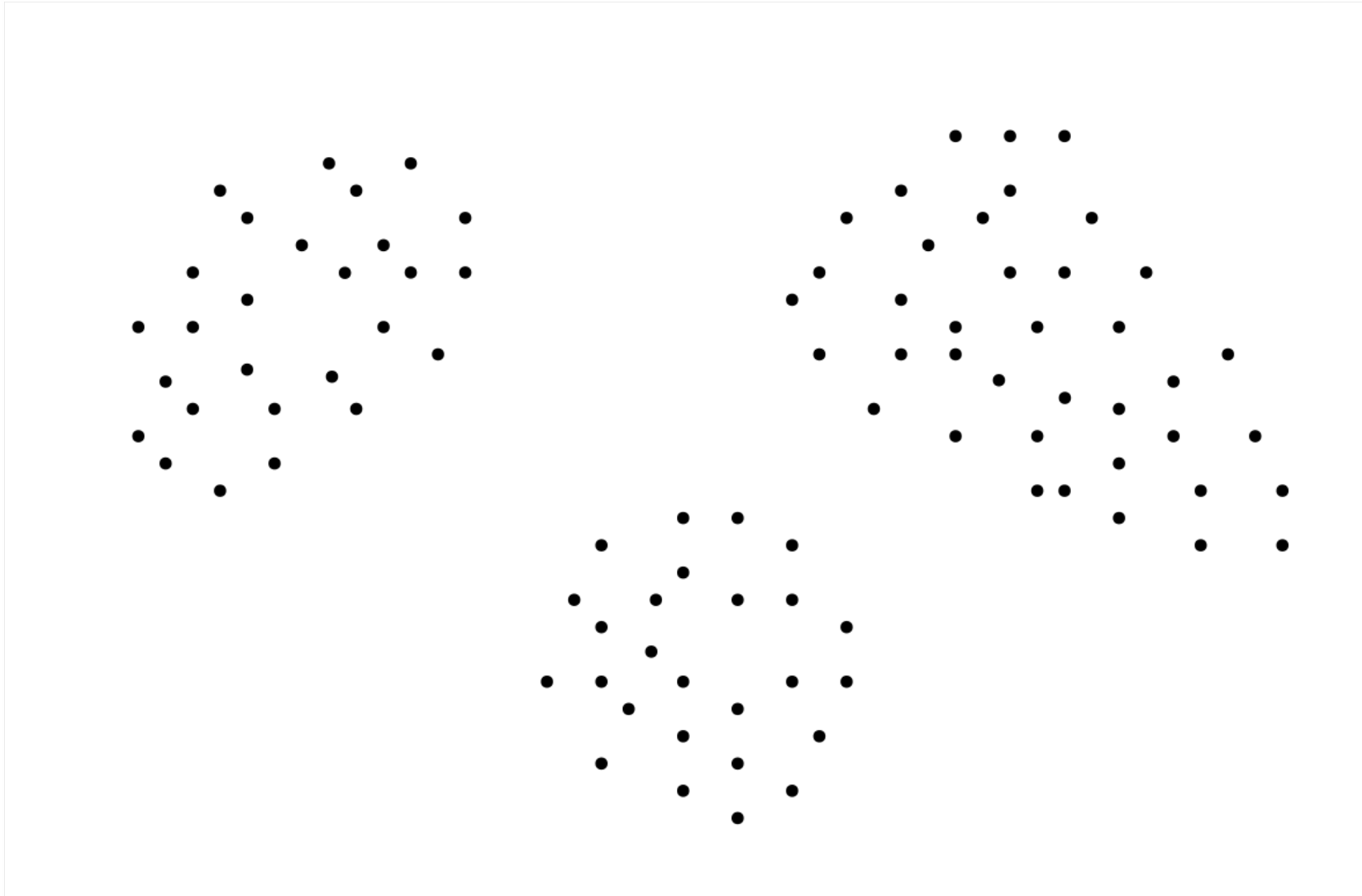
Advantage: Robust for data that is not well-distributed.

Disadvantage: Relatively costly with respect to running time.
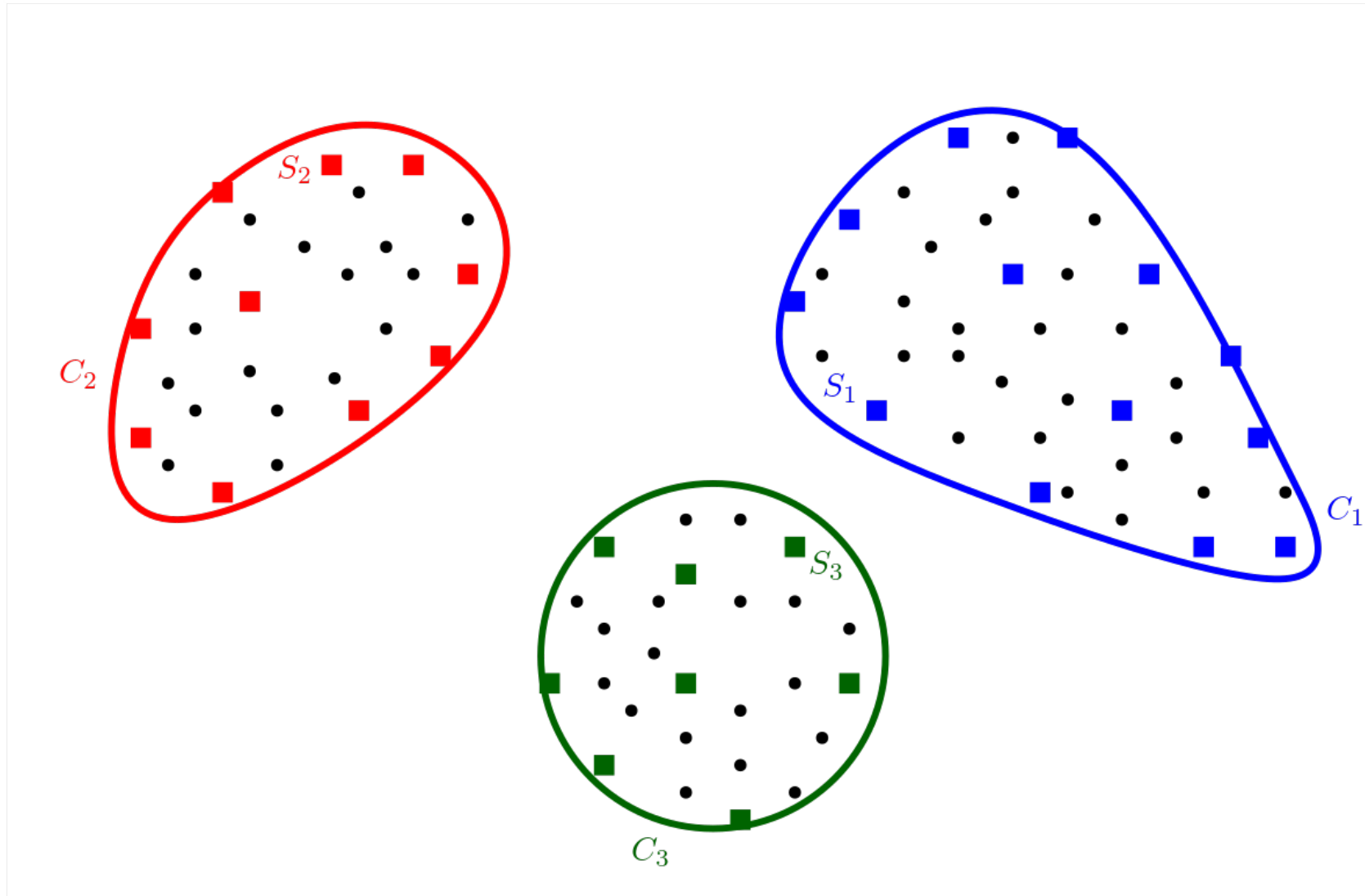
CURE (Clustering Using REpresentatives) algorithm
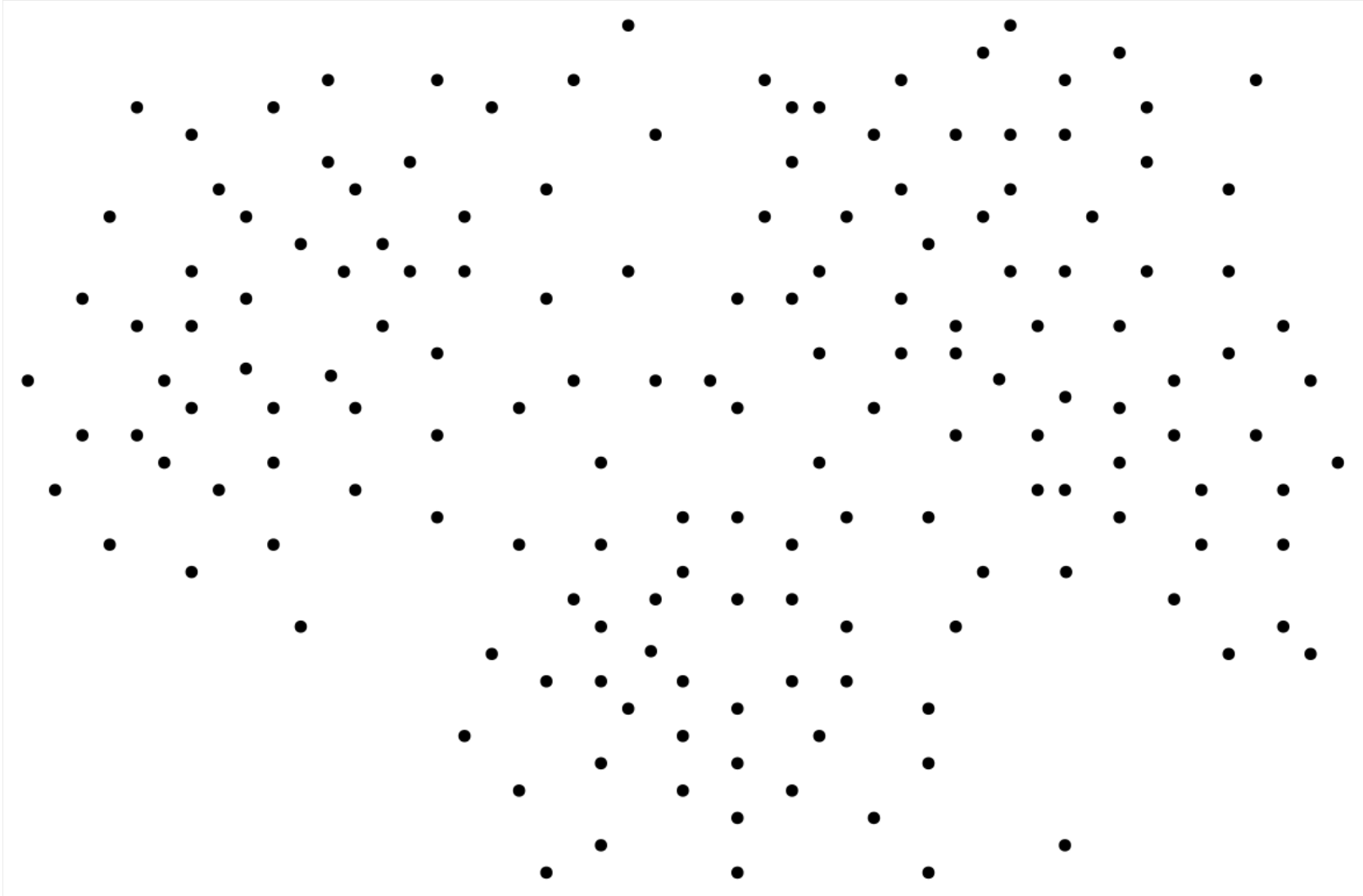
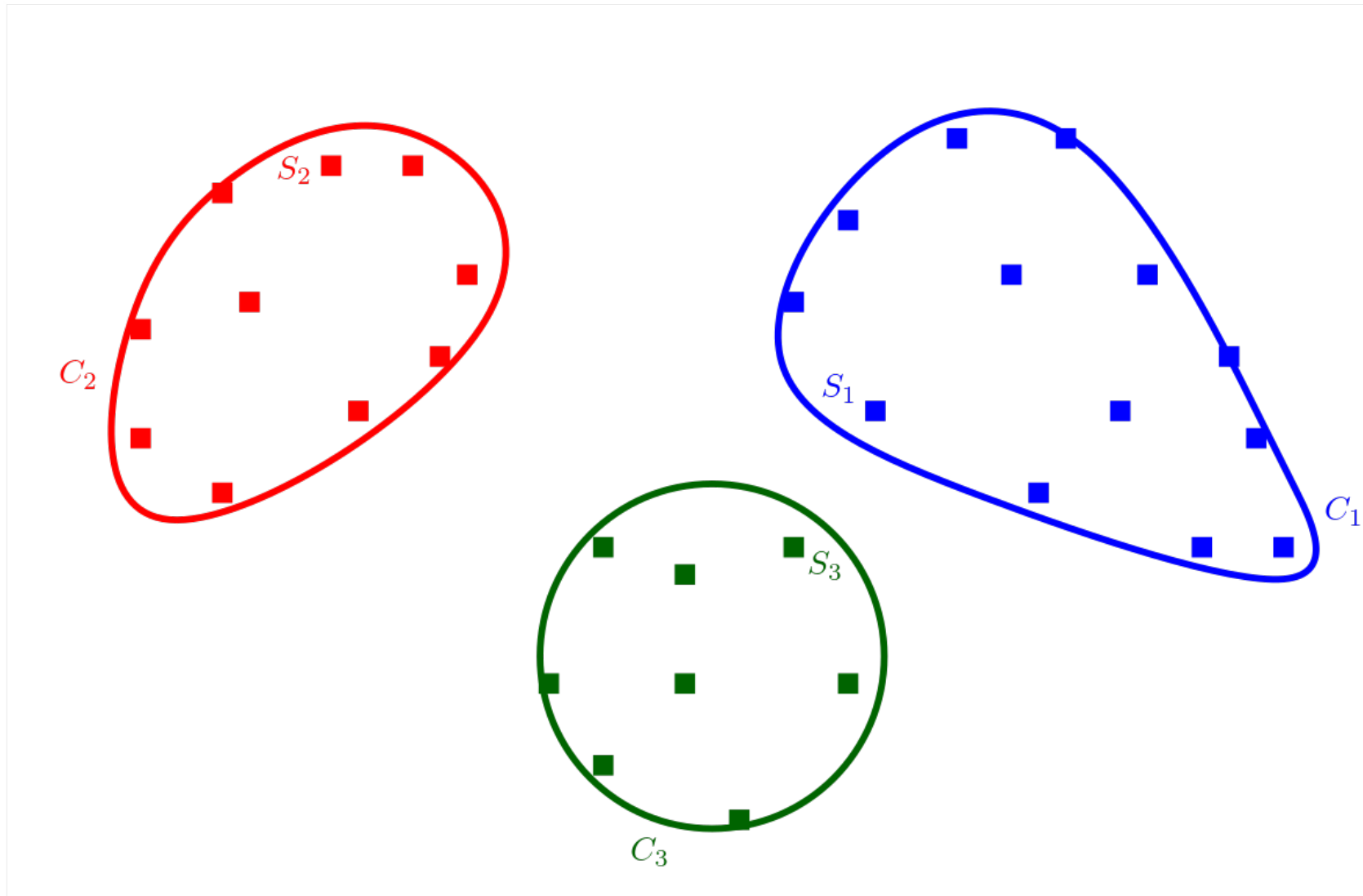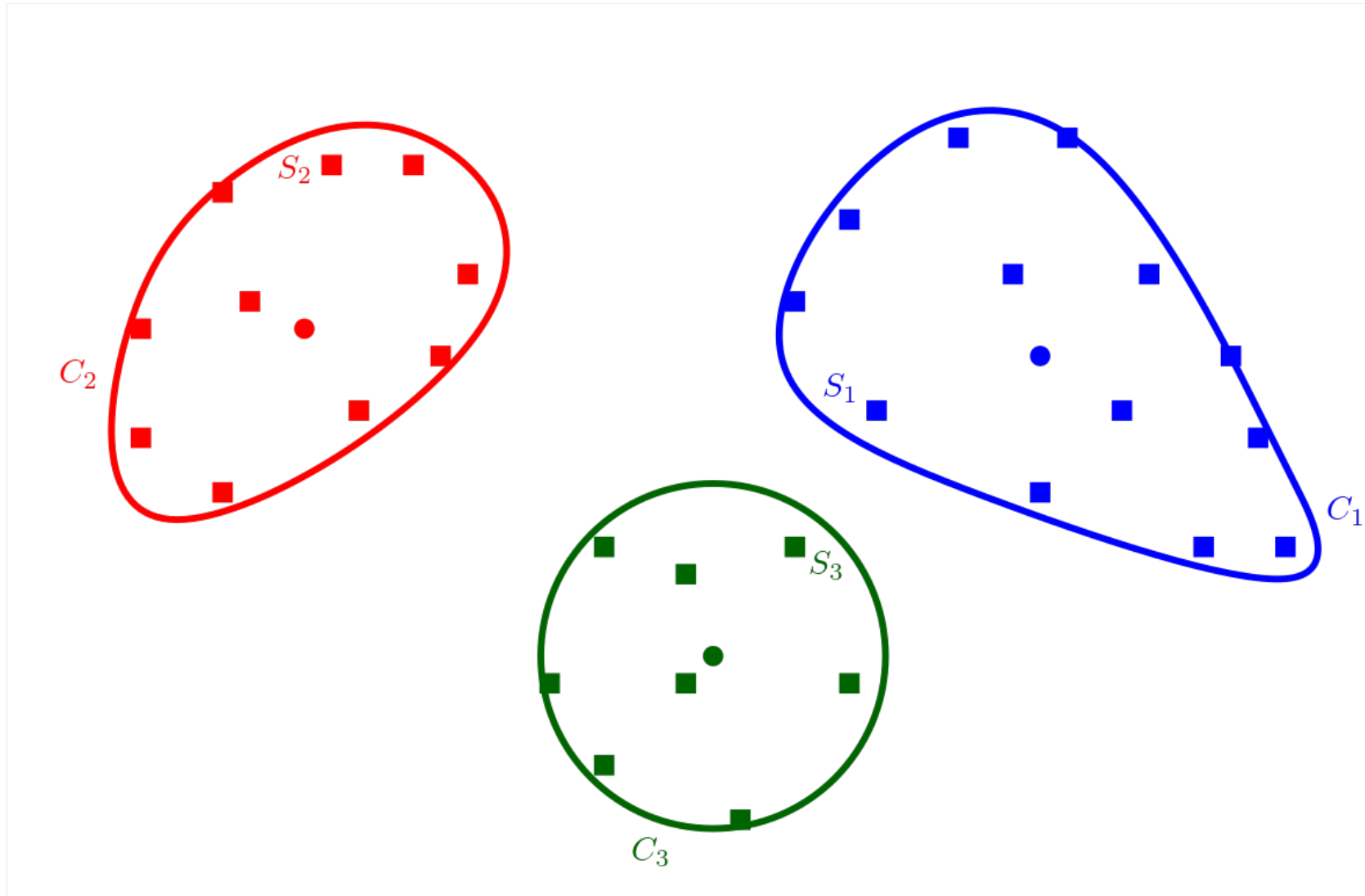# CURE (Clustering Using REpresentatives) algorithm

CURE (Clustering Using REpresentatives) algorithm

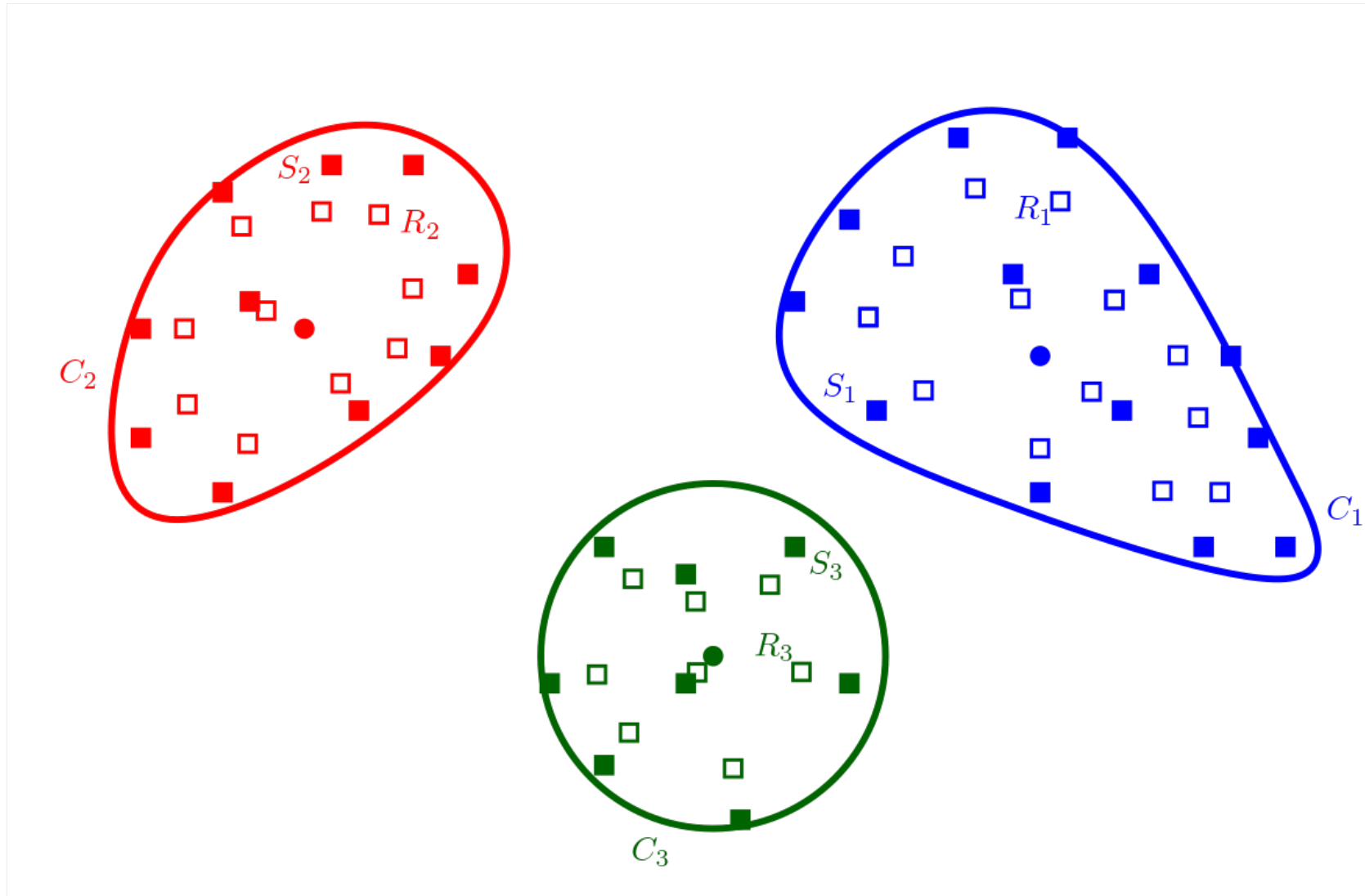# CURE (Clustering Using REpresentatives) algorithm

CURE (Clustering Using REpresentatives) algorithm

CURE (Clustering Using REpresentatives) algorithm

CURE (Clustering Using REpresentatives) algorithm

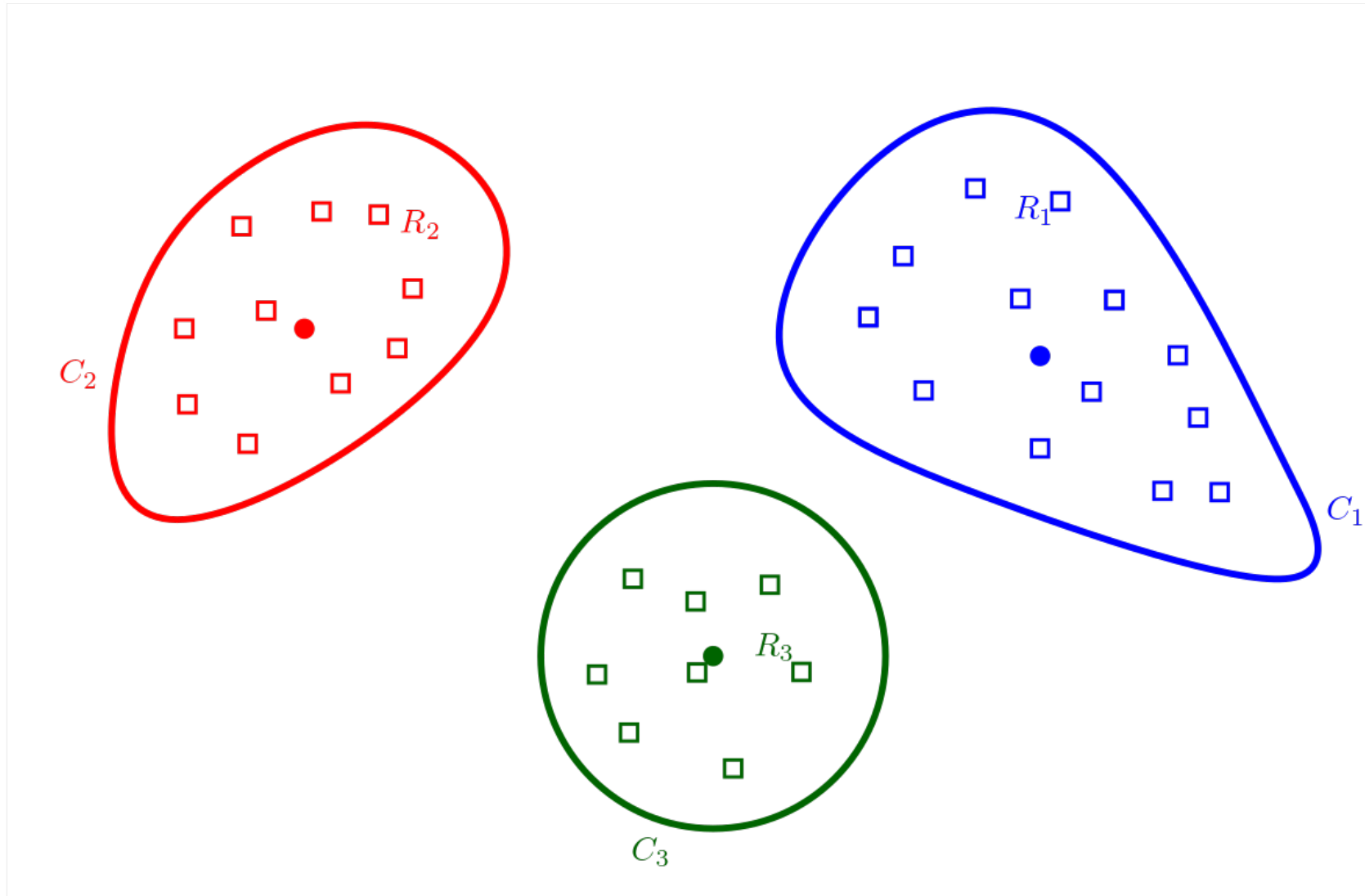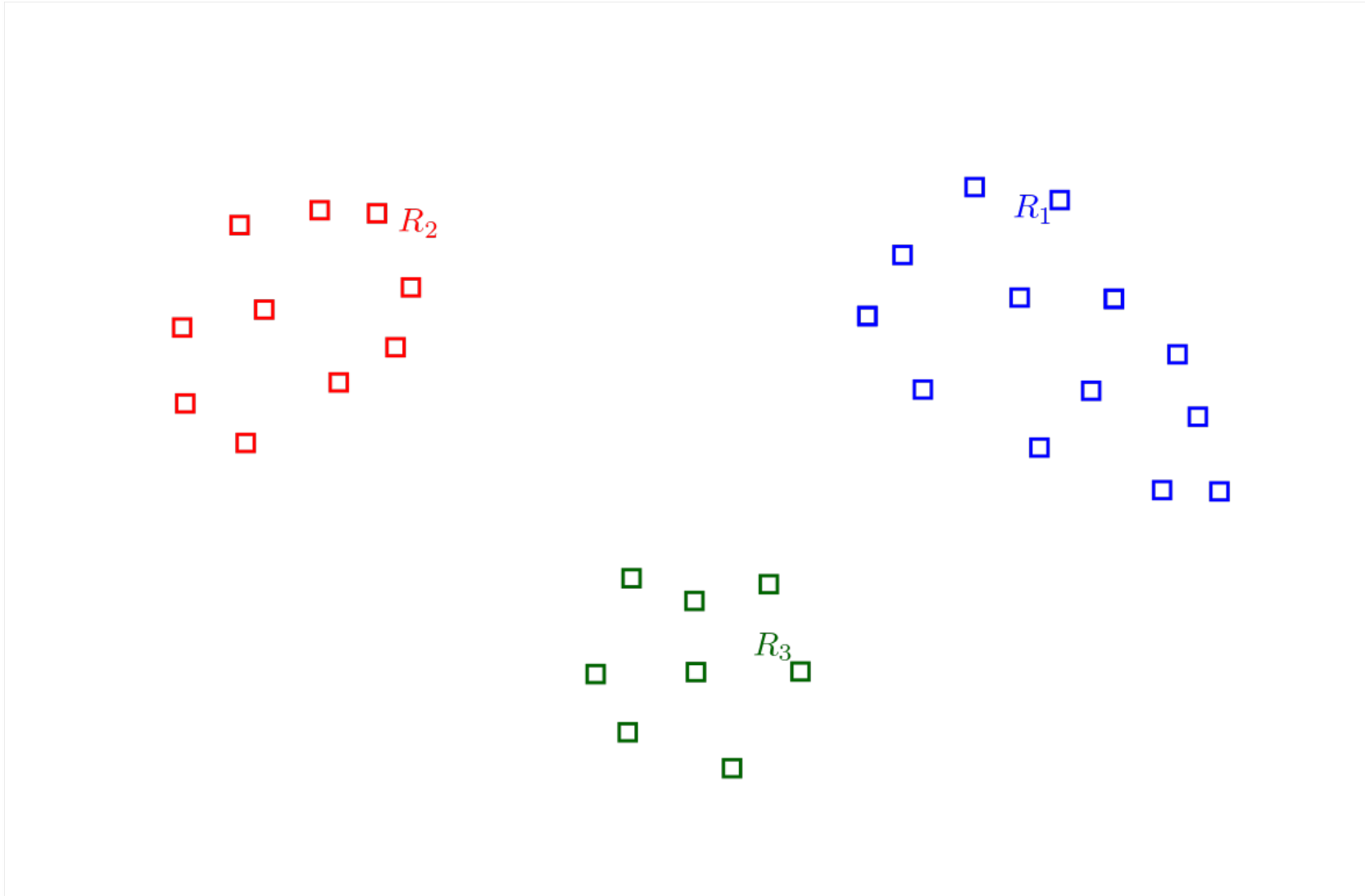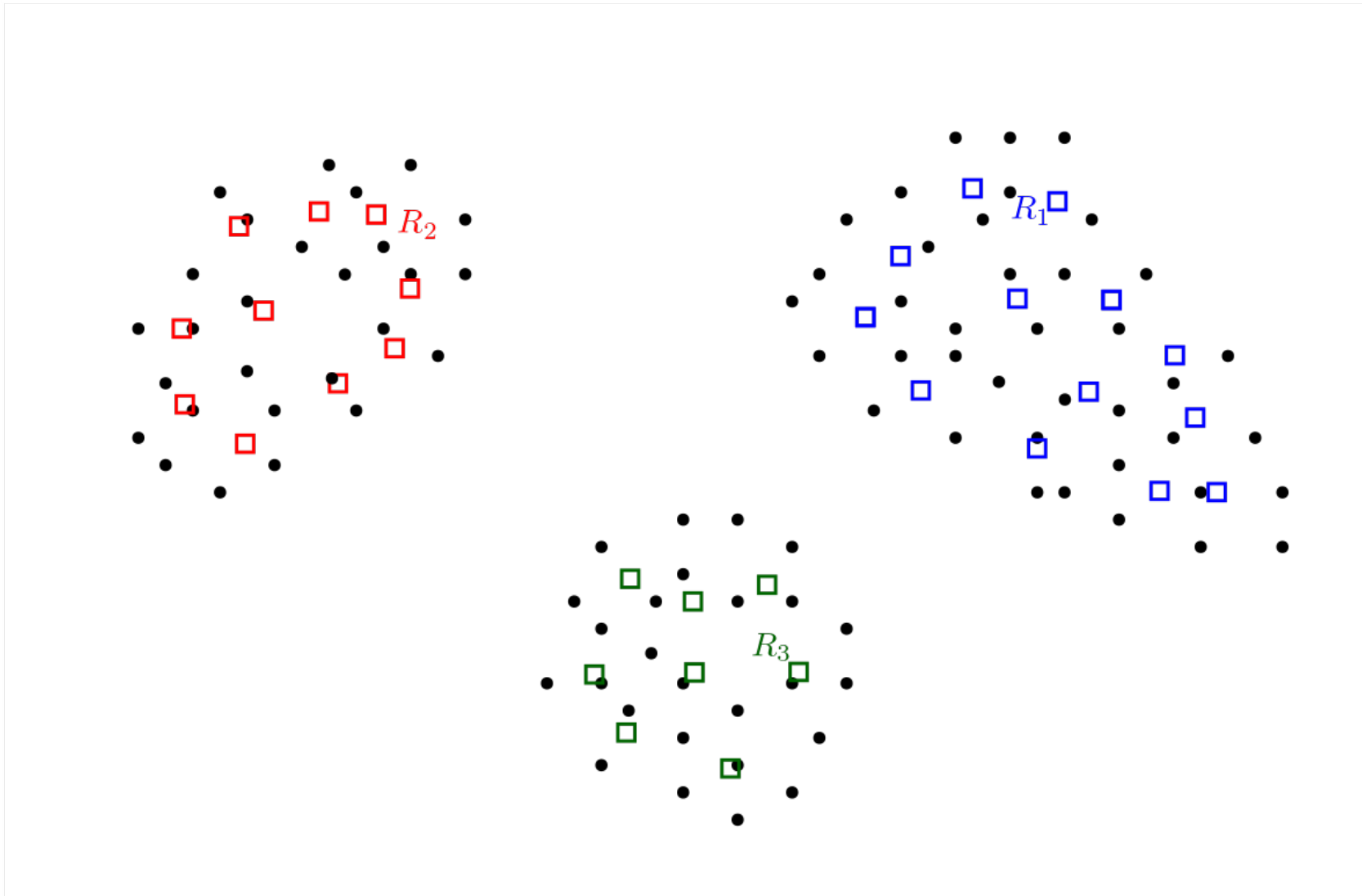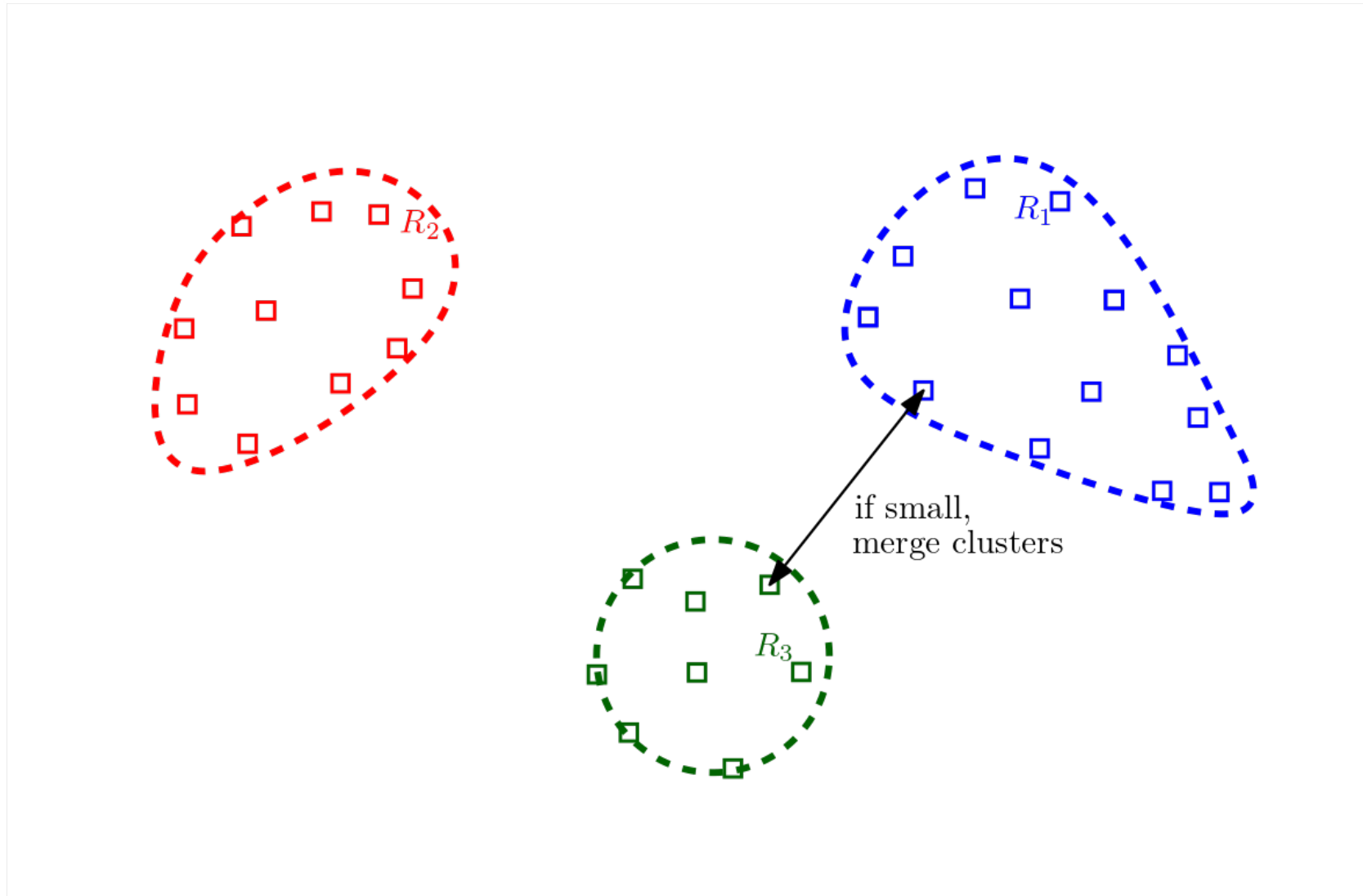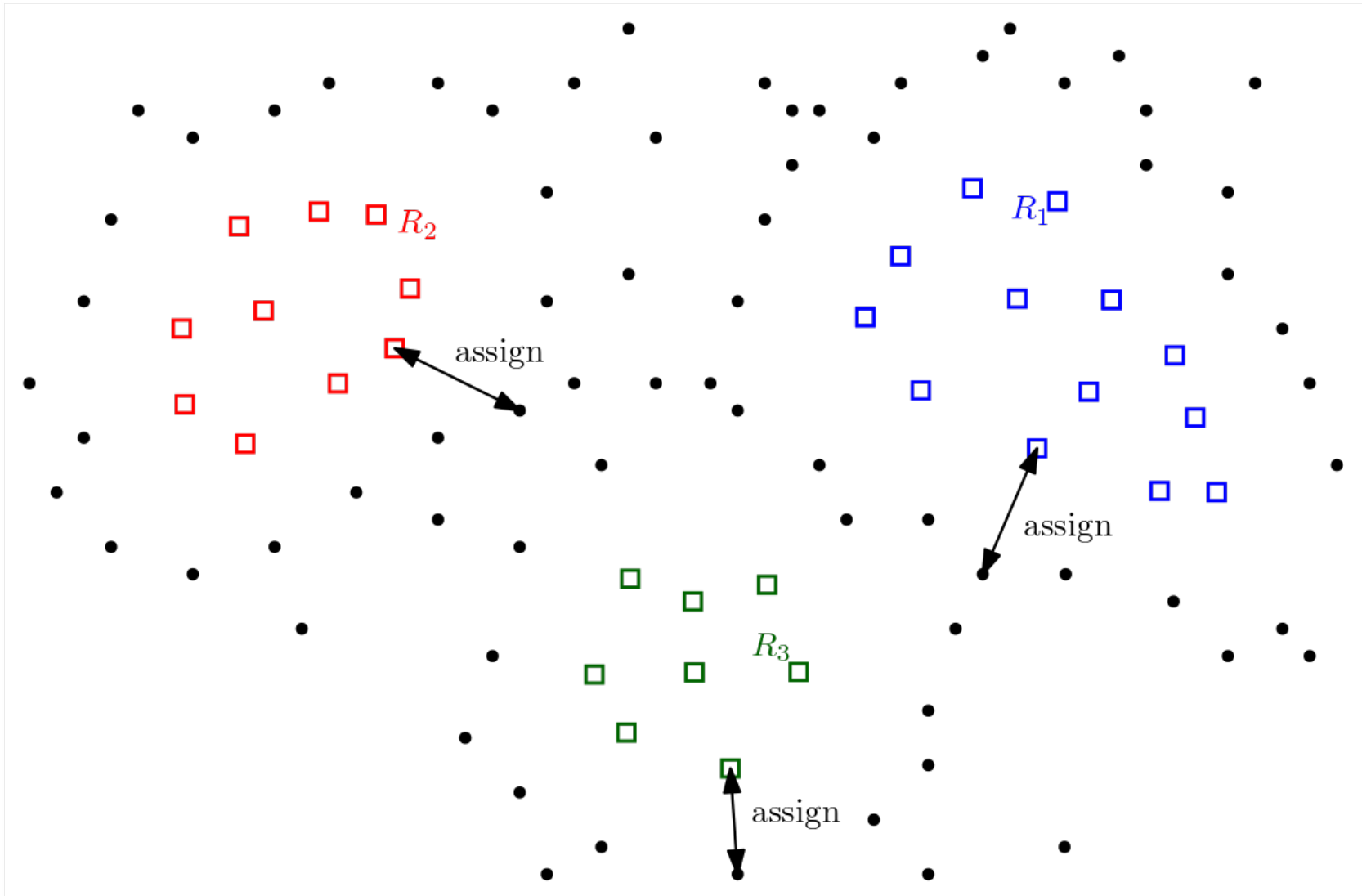# CURE (Clustering Using REpresentatives) algorithm

CURE (Clustering Using REpresentatives) algorithm

CURE (Clustering Using REpresentatives) algorithm

CURE (Clustering Using REpresentatives) algorithm

R_2

R_1

R_3

if small,
merge clusters

CURE (Clustering Using REpresentatives) algorithm

CURE (Clustering Using REpresentatives) algorithm

Original CURE algorithm

- Perform hierarchical clustering.

- In each step, (re)compute representatives for each cluster.

- Merge two clusters w.r.t. distance between representatives.

- Stop if desired number of clusters is reached (or other abort condition).

- Running time: $O(n^2 \log(n))$

DBSCAN algorithm

Name: **Density-based spatial clustering of applications with noise**

Parameters: $\varepsilon \in \mathbb{R}, m \in \mathbb{N}$.

Set S of all data points is divided into some different point categories:

- $p \in S$ is a **core point** if $|\{x \in S \mid d(p, x) \leq \varepsilon\}| \geq m$.

DBSCAN algorithm

Name: **Density-based spatial clustering of applications with noise**

Parameters: $\varepsilon \in \mathbb{R}, m \in \mathbb{N}$.

Set S of all data points is divided into some different point categories:

- $p \in S$ is a **core point** if $|\{x \in S \mid d(p, x) \leq \varepsilon\}| \geq m$.
- $x \in S$ is **directly reachable** from $p \in S$ if $d(p, x) \leq \varepsilon$ and p is a core point.

DBSCAN algorithm

Name: **Density-based spatial clustering of applications with noise**

Parameters: $\varepsilon \in \mathbb{R}, m \in \mathbb{N}$.

Set S of all data points is divided into some different point categories:

- $p \in S$ is a **core point** if $|\{x \in S \mid d(p,x) \leq \varepsilon\}| \geq m$.
- $x \in S$ is **directly reachable** from $p \in S$ if $d(p,x) \leq \varepsilon$ and p is a core point.
- $x \in S$ is **reachable** from $p \in S$ if there are points $x_1, \dots, x_k \in S$ such that:
  - $x_1 = p$ and $x_k = x$,
  - x_{i+1} is directly reachable from x_i for all $i \in \{1, \dots, k-1\}$,
  - $x_1, \dots, x_{k-1}$ are core points.

A reachable point that is no core point is called a **rim point**

DBSCAN algorithm

Name: **Density-based spatial clustering of applications with noise**

Parameters: $\varepsilon \in \mathbb{R}, m \in \mathbb{N}$.

Set S of all data points is divided into some different point categories:

- $p \in S$ is a **core point** if $|\{x \in S \mid d(p,x) \leq \varepsilon\}| \geq m$.
- $x \in S$ is **directly reachable** from $p \in S$ if $d(p,x) \leq \varepsilon$ and p is a core point.
- $x \in S$ is **reachable** from $p \in S$ if there are points $x_1, \ldots, x_k \in S$ such that:
    - $x_1 = p$ and $x_k = x$,
    - x_{i+1} is directly reachable from x_i for all $i \in \{1, \ldots, k-1\}$,
    - $x_1, \ldots, x_{k-1}$ are core points.

A reachable point that is no core point is called a **rim point**.

- $x \in S$ is an **outlier (or noise) point** if it is not reachable from any $p \in S$.

DBSCAN algorithm

Initialise: Mark each point as unvisited.

DBSCAN algorithm

Initialise: Mark each point as unvisited.

Check each point p:

- if unvisited:
    - if p is not core, mark as visited (rim point or outlier).

DBSCAN algorithm

Initialise: Mark each point as unvisited.

Check each point p:

- if unvisited:

  - if p is not core, mark as visited (rim point or outlier).

  - if p is core, define cluster:

  $$C := \{x \mid x \ is \ reachable \ from \ p \ and \ x \ is \ not \ already \ in \ a \ cluster\}.$$

DBSCAN algorithm

Initialise: Mark each point as unvisited.

Check each point p:

• if unvisited:

  • if p is not core, mark as visited (rim point or outlier).

  • if p is core, define cluster:

  $C := \{x \mid x\ is\ reachable\ from\ p\ and\ x\ is\ not\ already\ in\ a\ cluster\}.$

Running time (worst-case): $O(n^2)$

Advantage: Can identify clusters whose shape is more irregular.

DBSCAN algorithm

Published by Karthik Ramasamy on Youtube. See this link.