Clustering and Dimensionality Reduction

Henning Petzka

Unsupervized Learning

Recap

- No label information given
- The task of learning is not prediction, but to **detect** underlying **structure**.
- The underlying structure can be thought of as learning a **representation** of the data

$$\phi : X \rightarrow \mathcal{F}$$

Clustering

Idea

- Clustering is an unsupervized learning problem
- Goal: **Assign** each data point to a **group** (of data points)
- Points within a group (intra-similarity) are more similar to points of other groups (inter-similarity).

Intra-similarity > Inter-similarity

Example: Assign these people to groups

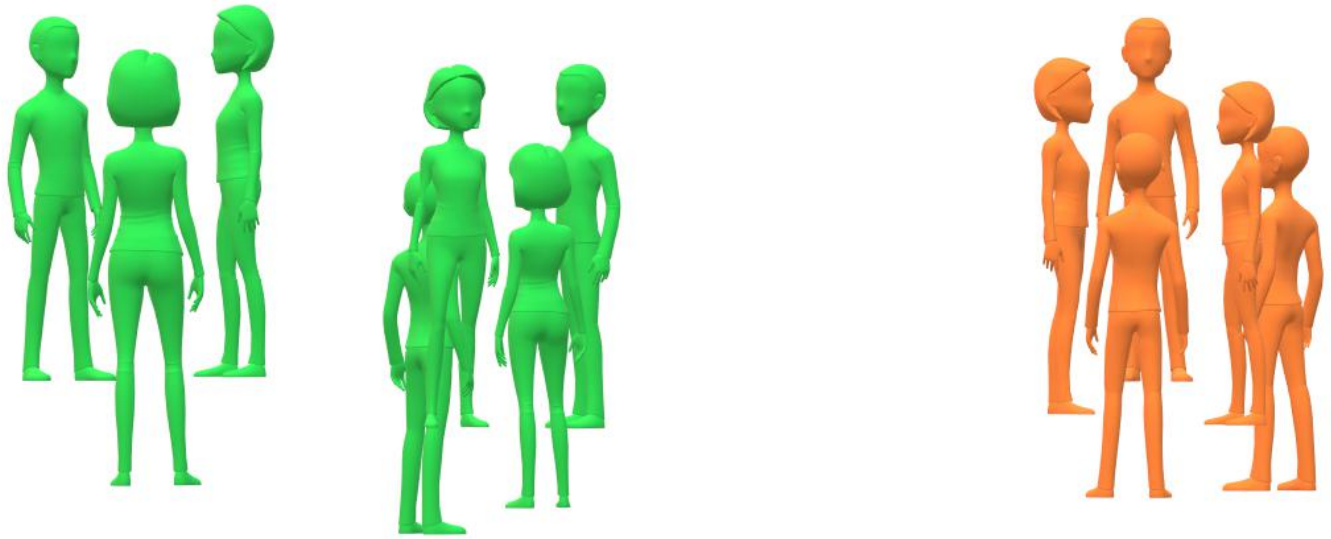
Clustering

Idea

- Clustering is an unsupervized learning problem
- Goal: **Assign** each data point to a **group** (of data points)
- Points within a group (intra-similarity) are more similar to points of other groups (inter-similarity).

Intra-similarity > Inter-similarity

Example: Assign these people to groups
Solution: 2 groups



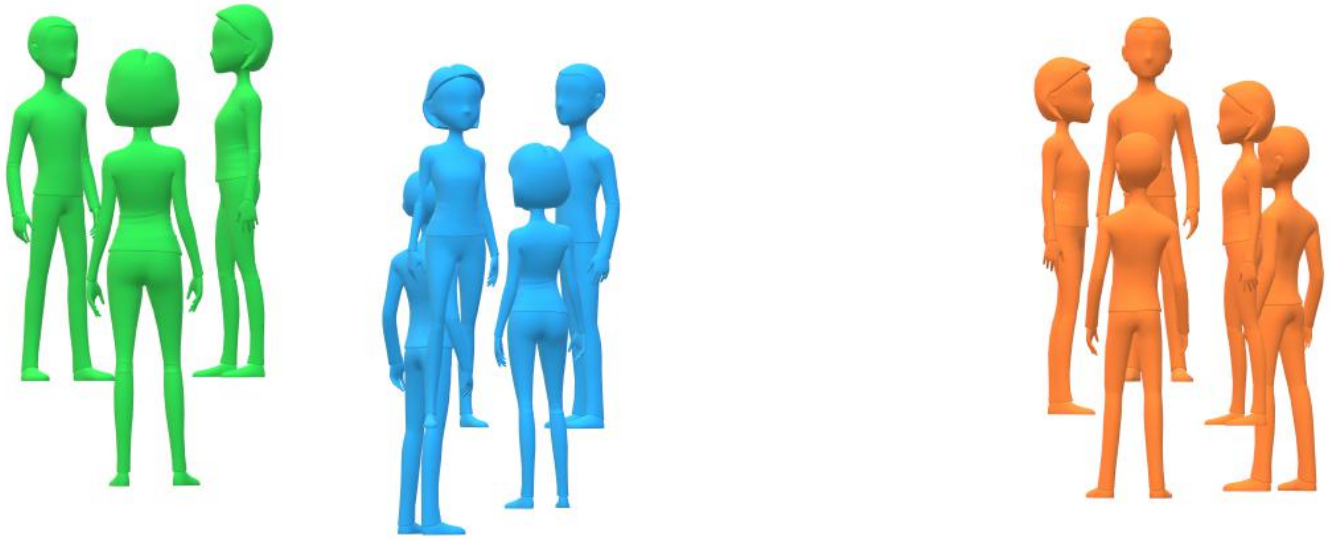
Clustering

Idea

- Clustering is an unsupervized learning problem
- Goal: **Assign** each data point to a **group** (of data points)
- Points within a group (intra-similarity) are more similar to points of other groups (inter-similarity).

Intra-similarity > Inter-similarity

Example: Assign these people to groups
Solution: 3 groups



Clustering
Applications

Market Segmentation

Group your customers into groups of interest

Biological network identification

Determine groups of proteins that belong to a common bilogical process

Social Network Analysis

Identify groups of friends or groups of people with shared opinions

Exploratory Data Mining

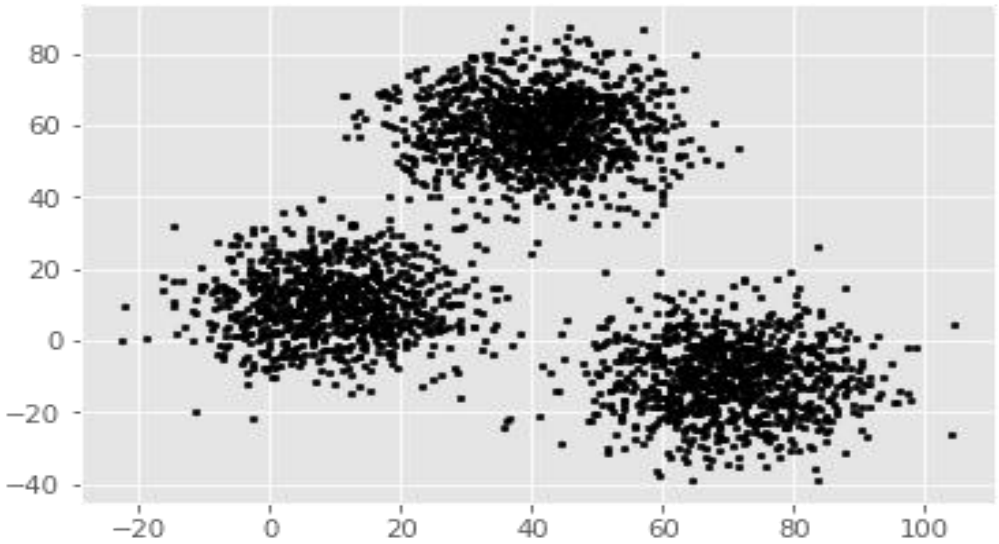
Get an overview of your data by investigating natural subset of data

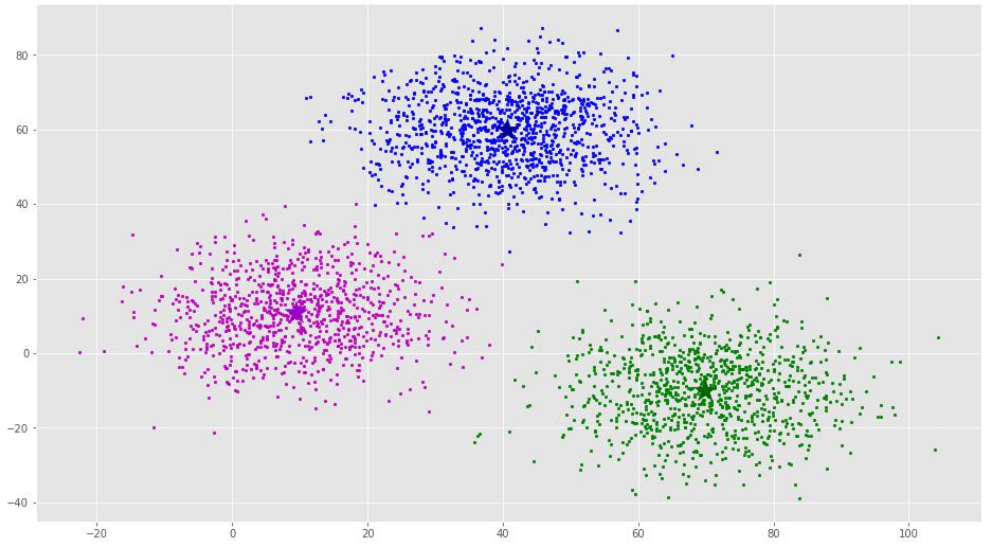
Clustering

K-Means

- **K-Means** is the most popular clustering technique
- Each cluster has a **centroid**
- Centroids define the learned model

- K = number of clusters: A parameter we need to choose

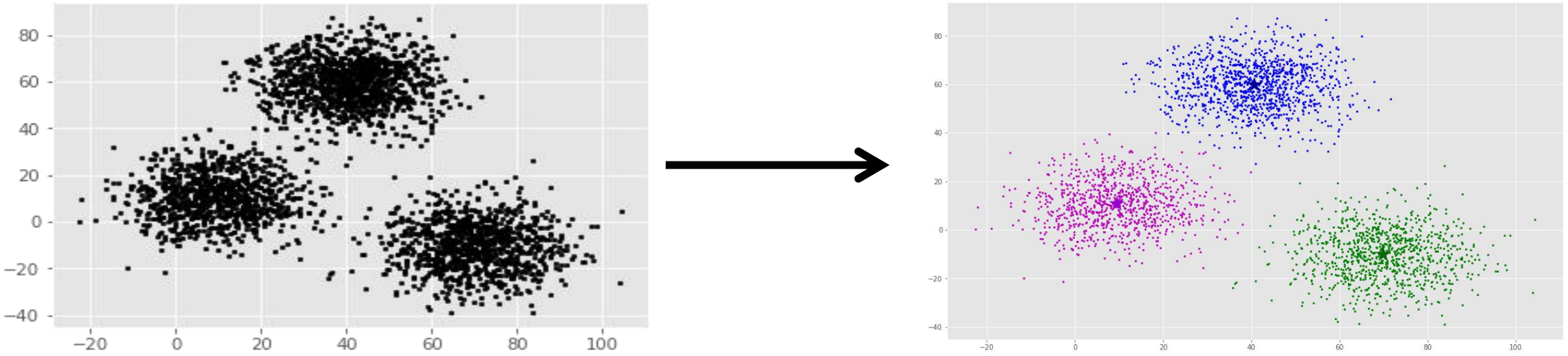




K-Means

How it works

- **Initialize K centroids** randomly
- K-Means algorithm consists of **two steps**
- These two steps are **repeated iteratively**



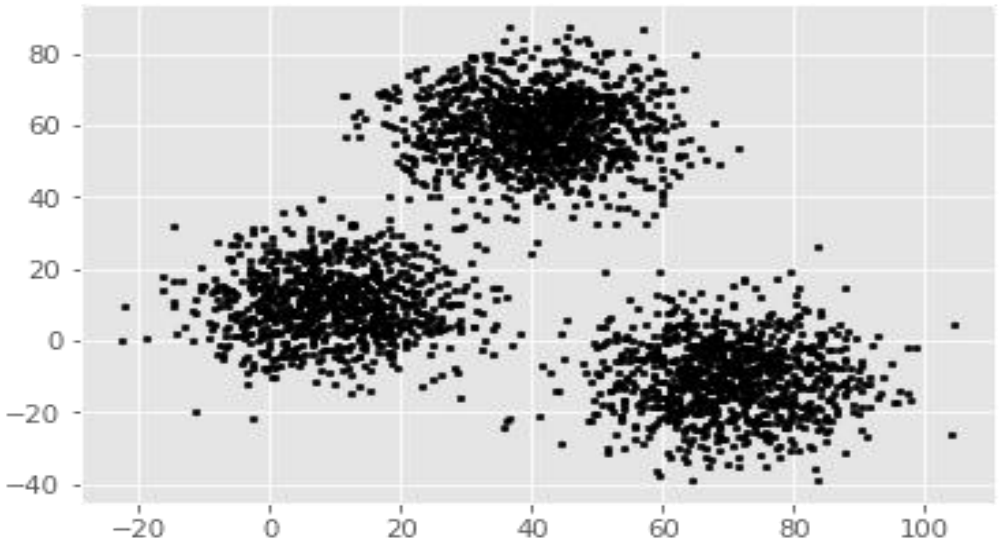
K-Means

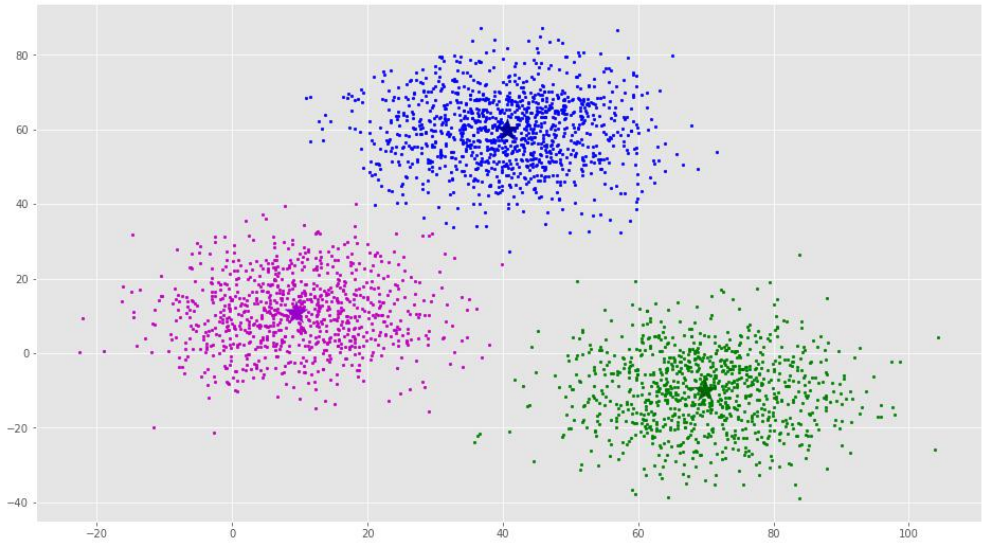
How it works

- **Initialize k centroids** randomly
- K-Means algorithm consists of **two steps**
- These two steps are **repeated iteratively**

A.) Assign each data point to its closest centroid

B.) Update each centroid by computing the **mean** of all points assigned to the centroid

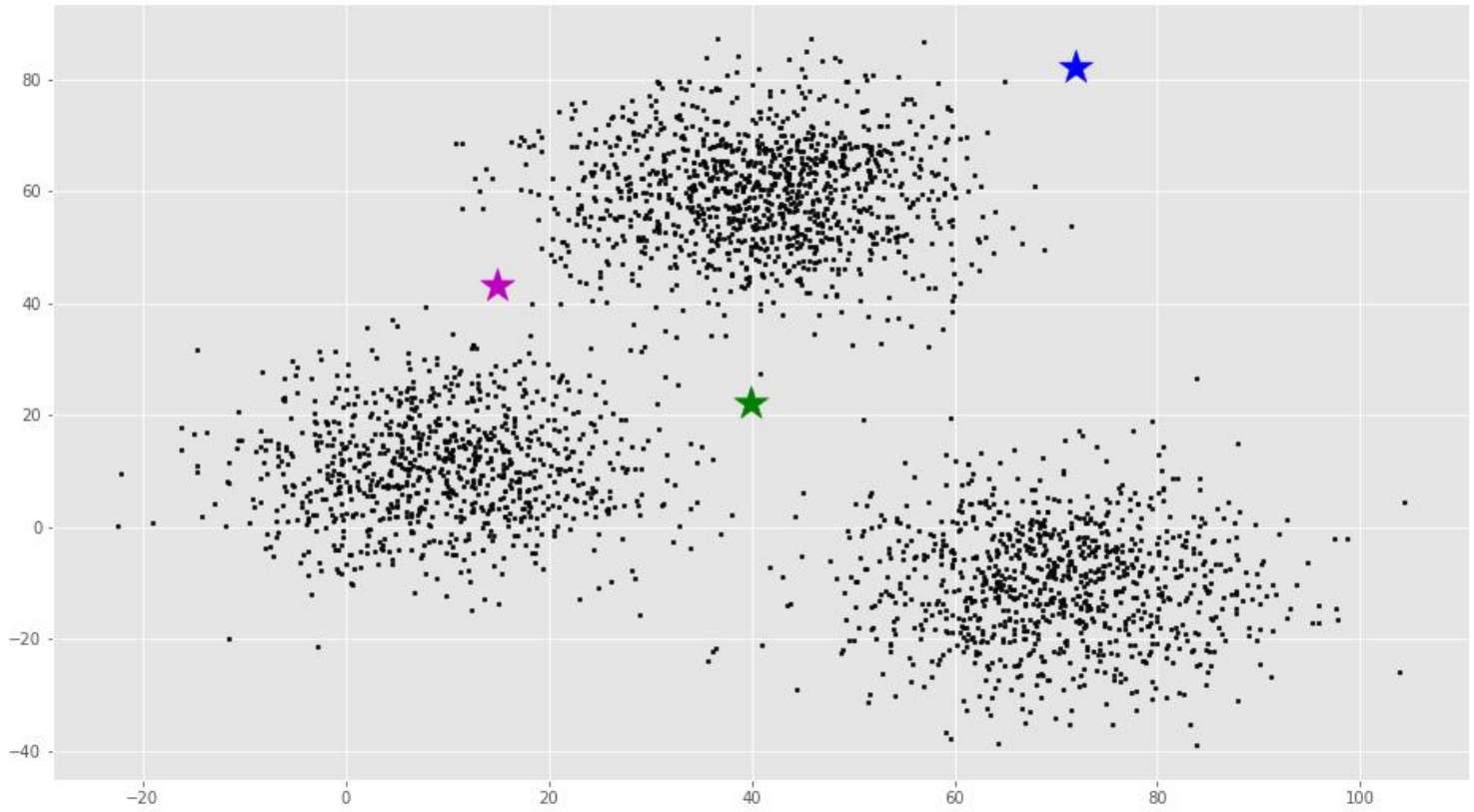




K-Means

How it works

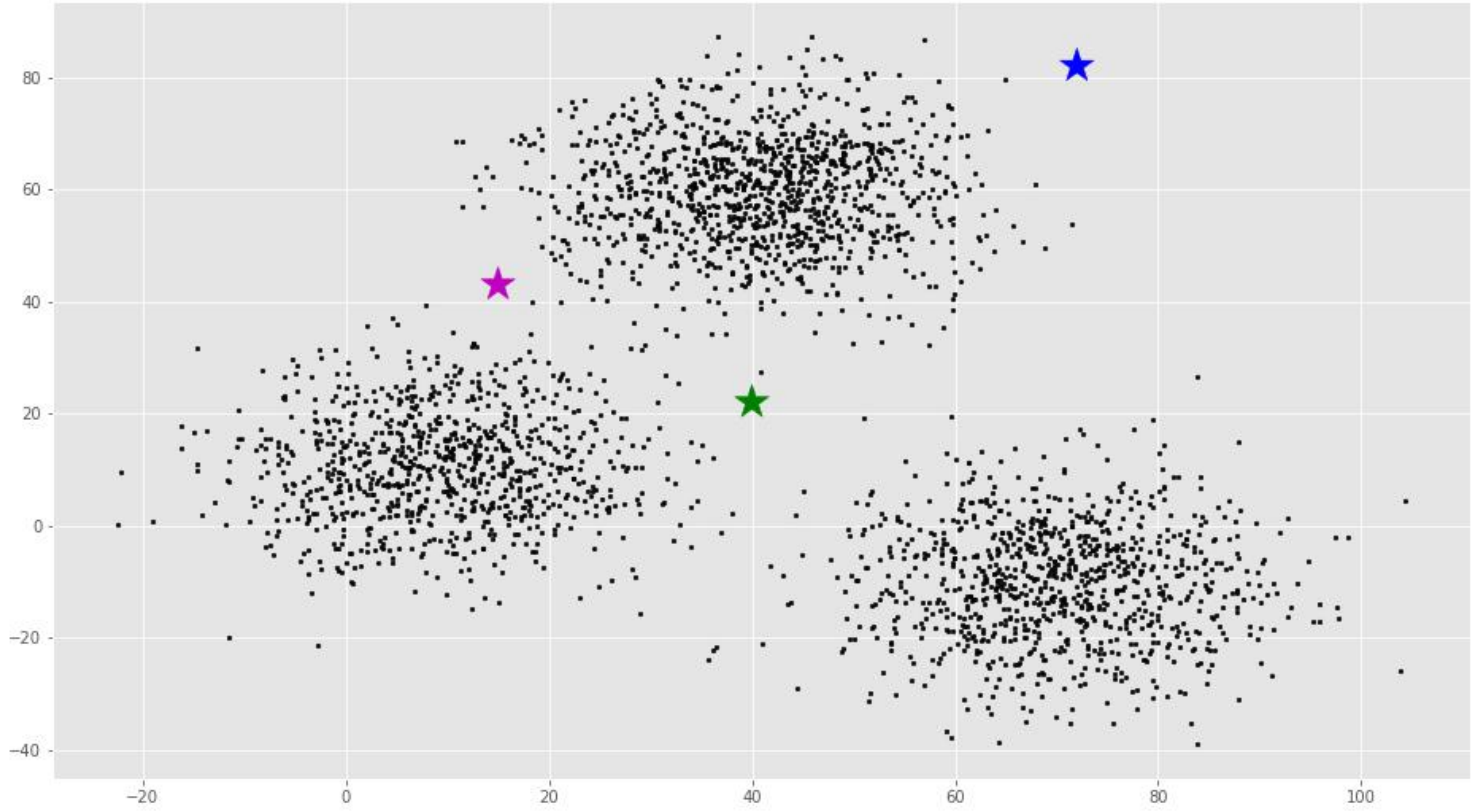
Step 0: Initialize centroids randomly



K-Means

How it works

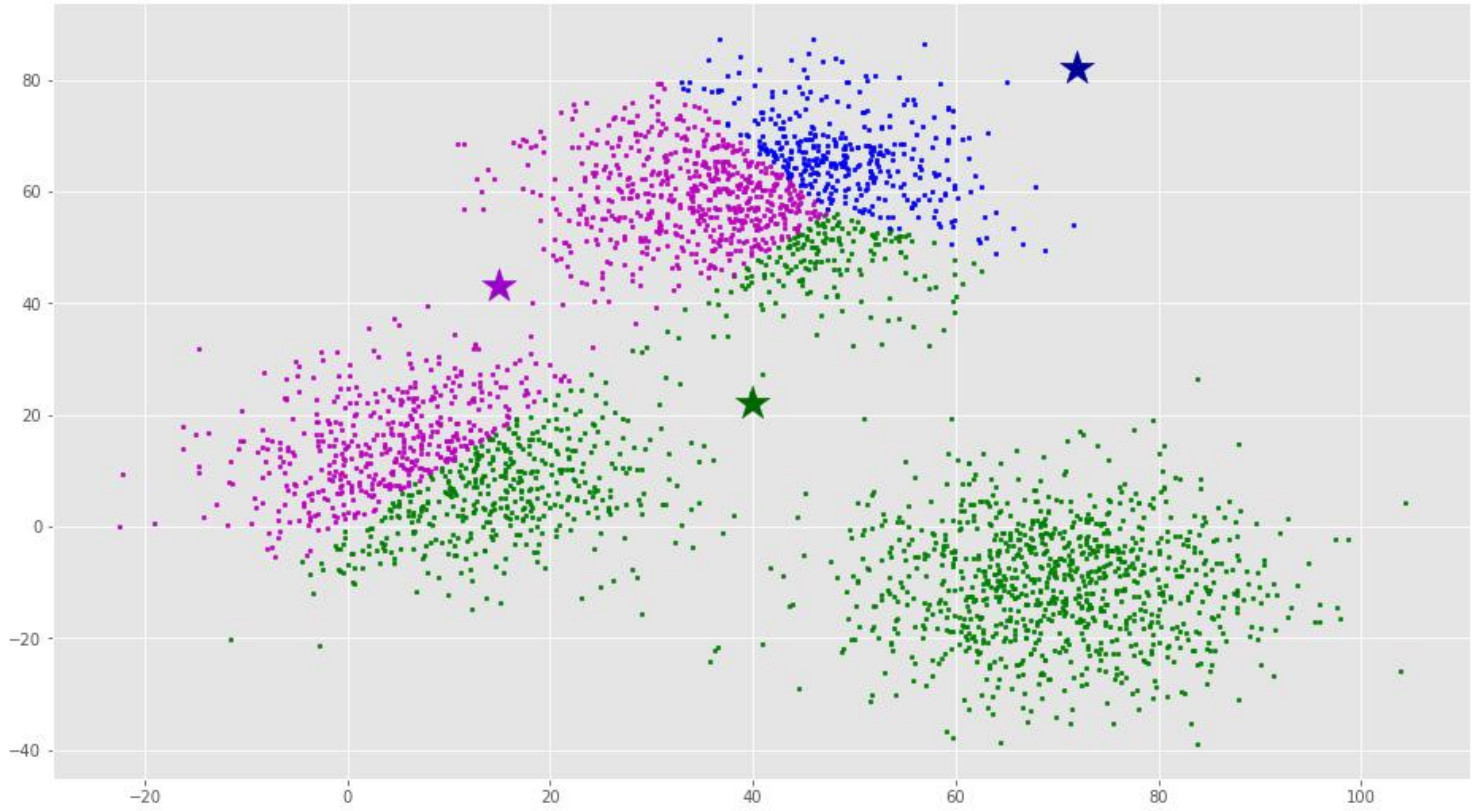
Step 1.A: Assign each data point to its closest centroid



K-Means

How it works

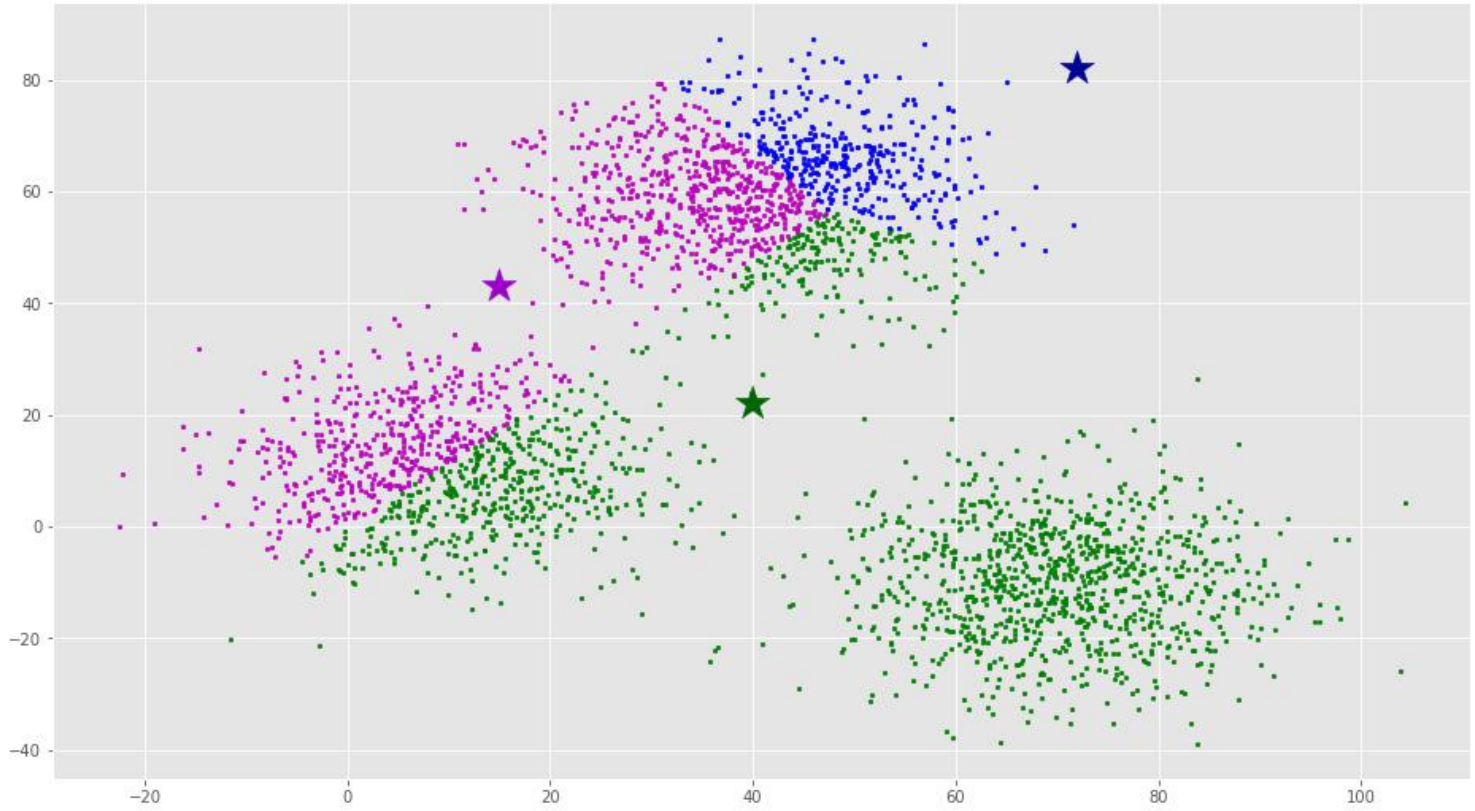
Step 1.A: Assign each data point to its closest centroid



K-Means

How it works

Step 1.B: Update each centroid by computing the **mean** of all points assigned to the centroid



K-Means

How it works

Step 1.B: Update each centroid by computing the **mean** of all points assigned to the centroid

K-Means

How it works

Step 2.A: Assign each data point to its closest centroid

K-Means

How it works

Step 2.A: Assign each data point to its closest centroid

K-Means

How it works

Step 2.B: Update each centroid by computing the **mean** of all points assigned to the centroid

K-Means

How it works

Step 2.B: Update each centroid by computing the **mean** of all points assigned to the centroid

K-Means

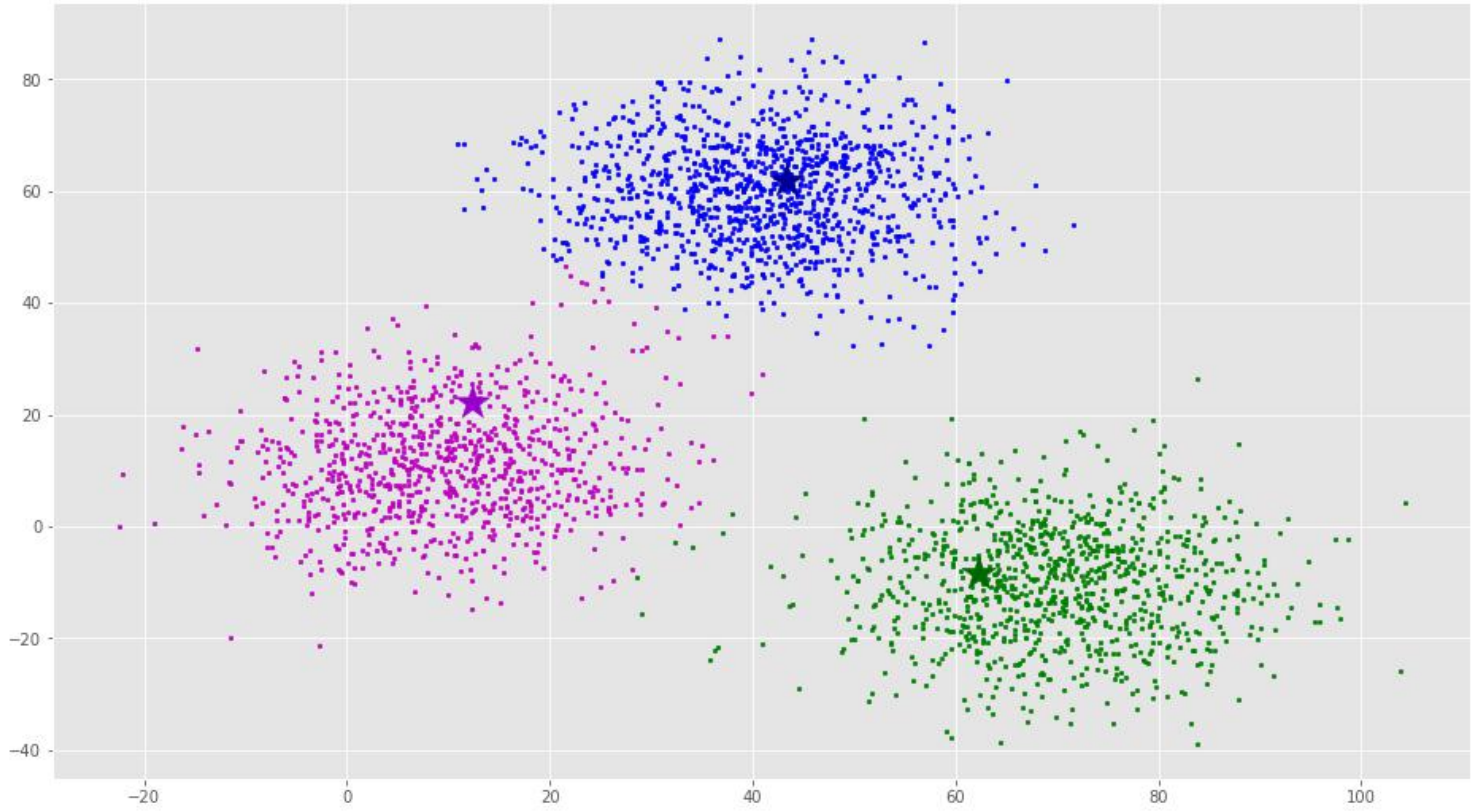
How it works

Step 3.A: Assign each data point to its closest centroid

K-Means

How it works

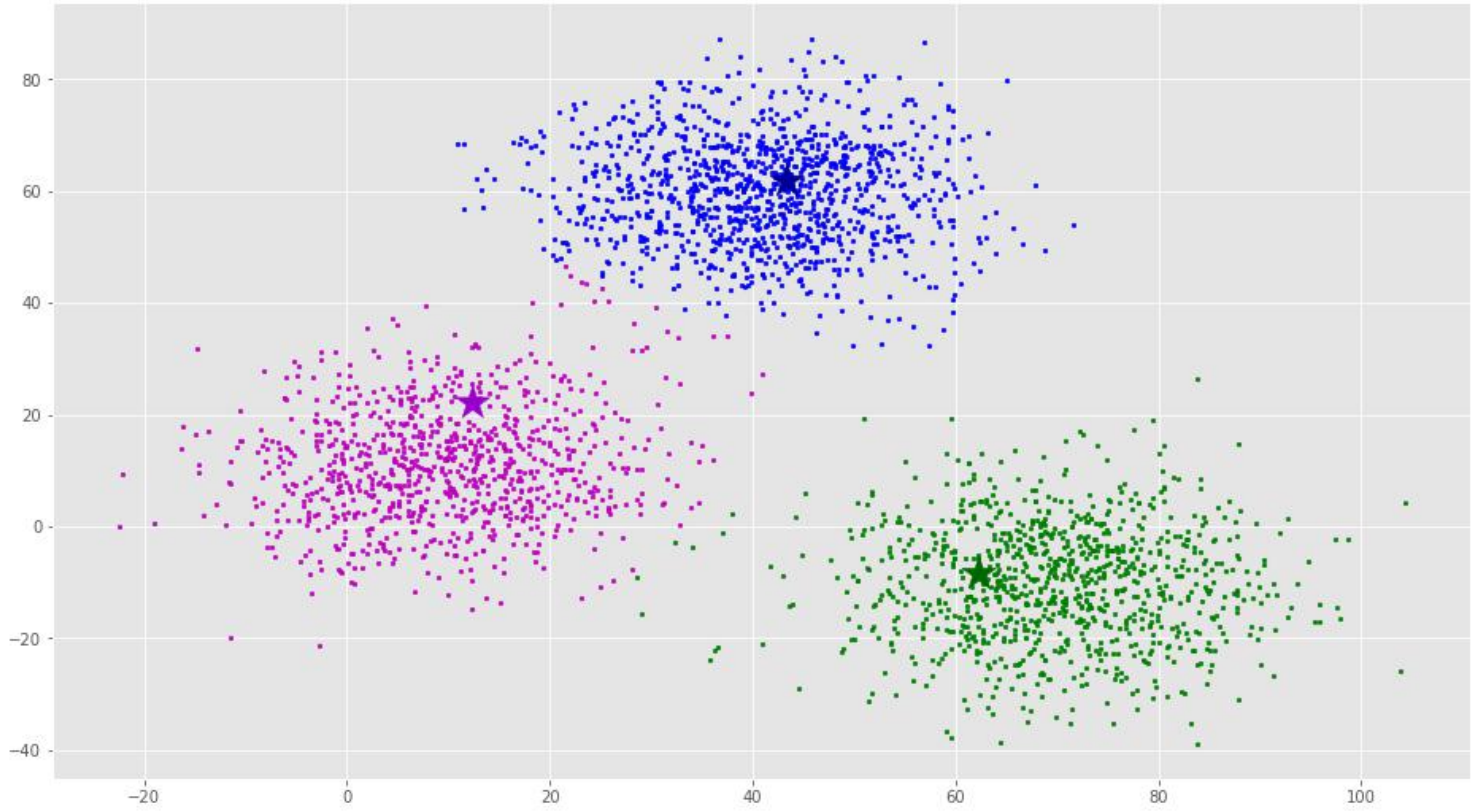
Step 3.A: Assign each data point to its closest centroid



K-Means

How it works

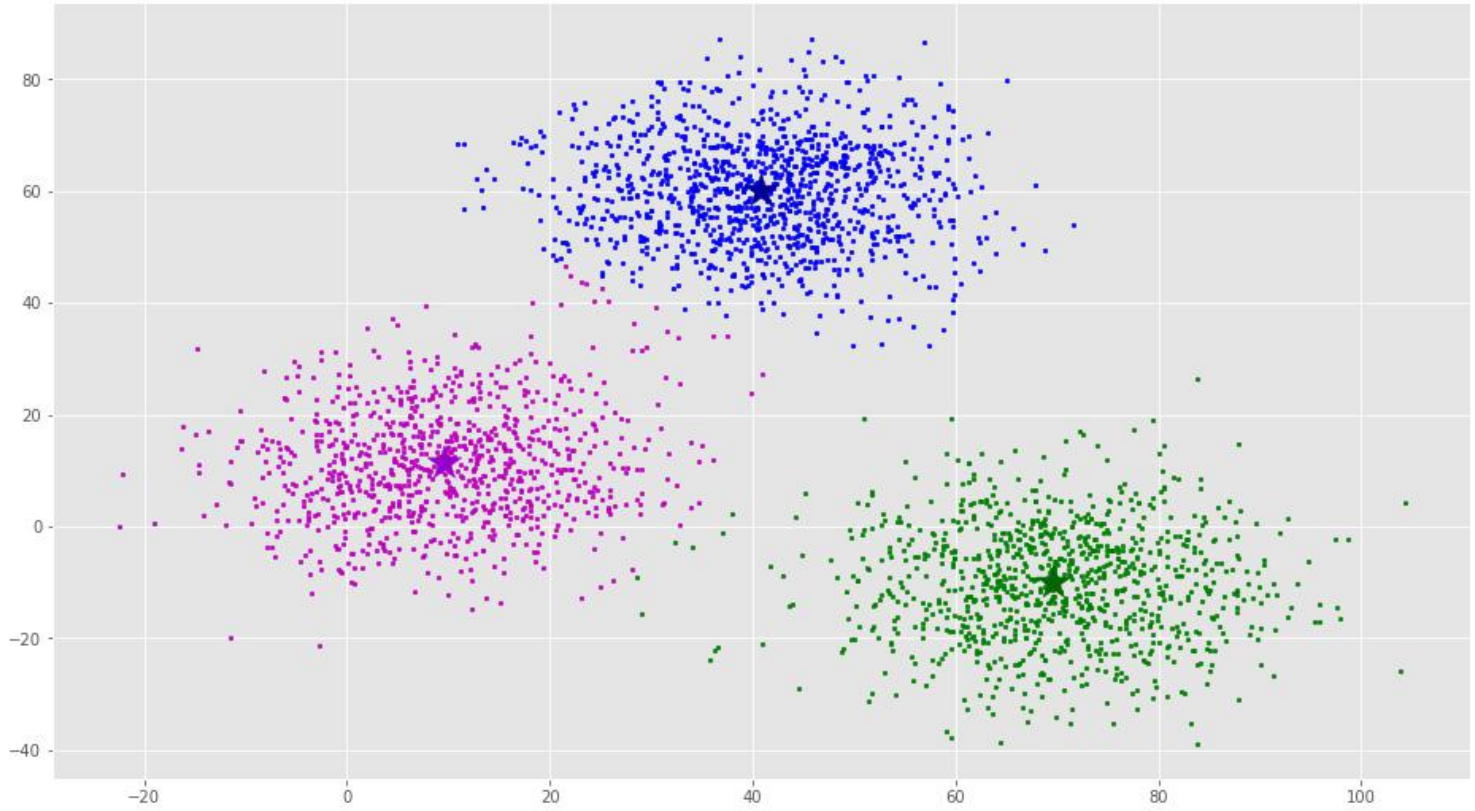
Step 3.B: Update each centroid by computing the **mean** of all points assigned to the centroid



K-Means

How it works

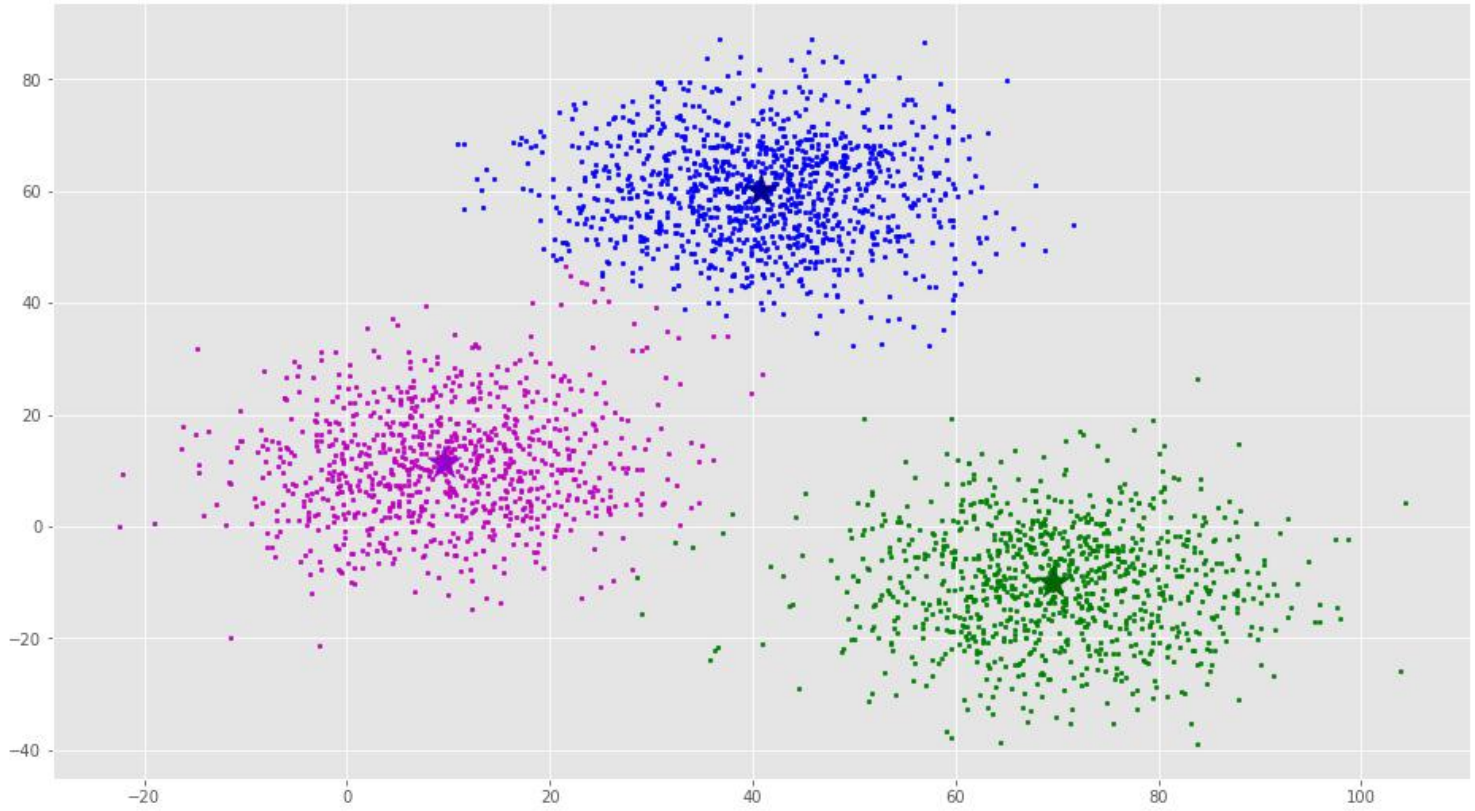
Step 3.B: Update each centroid by computing the **mean** of all points assigned to the centroid



K-Means

How it works

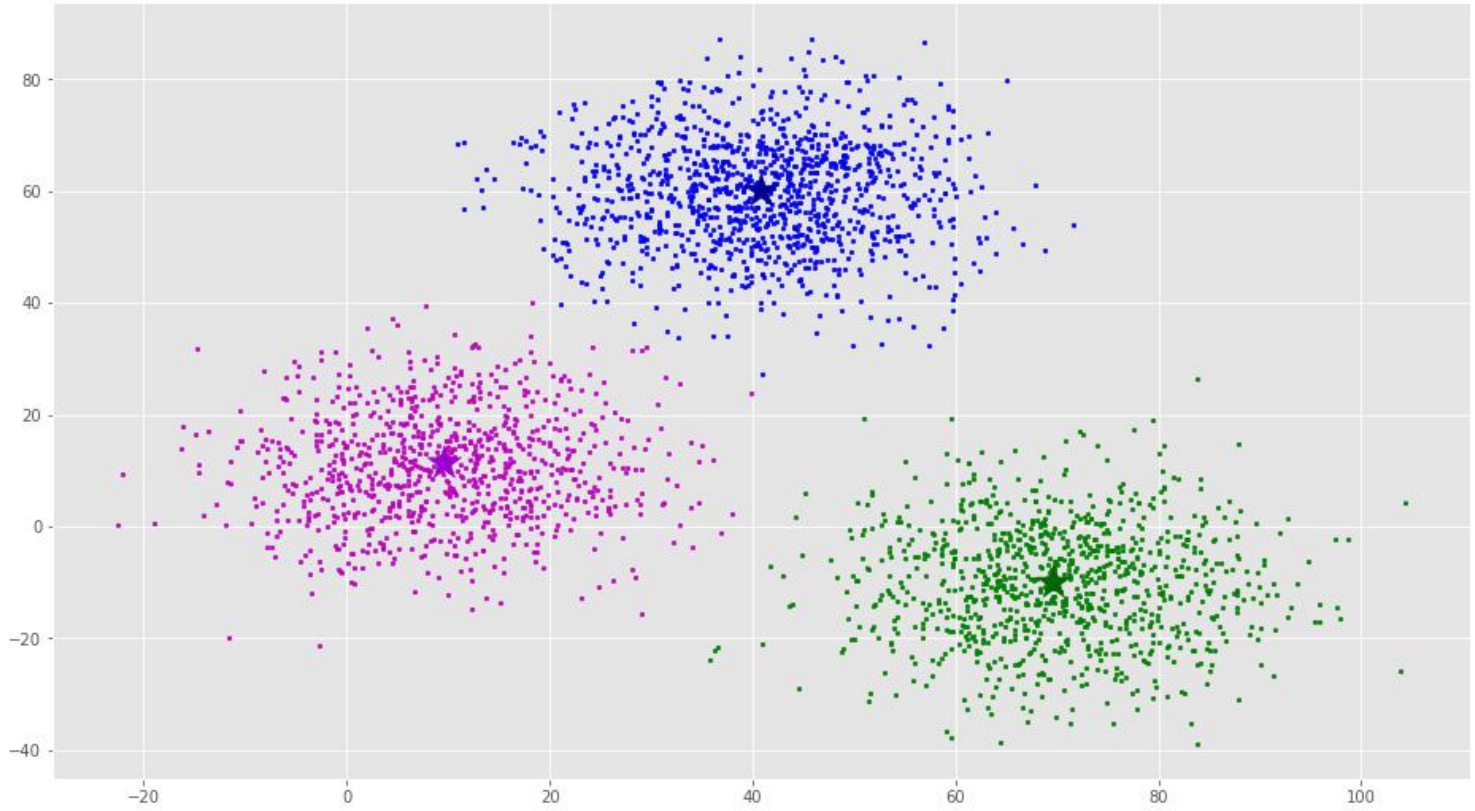
Step 4.A: Assign each data point to its closest centroid



K-Means

How it works

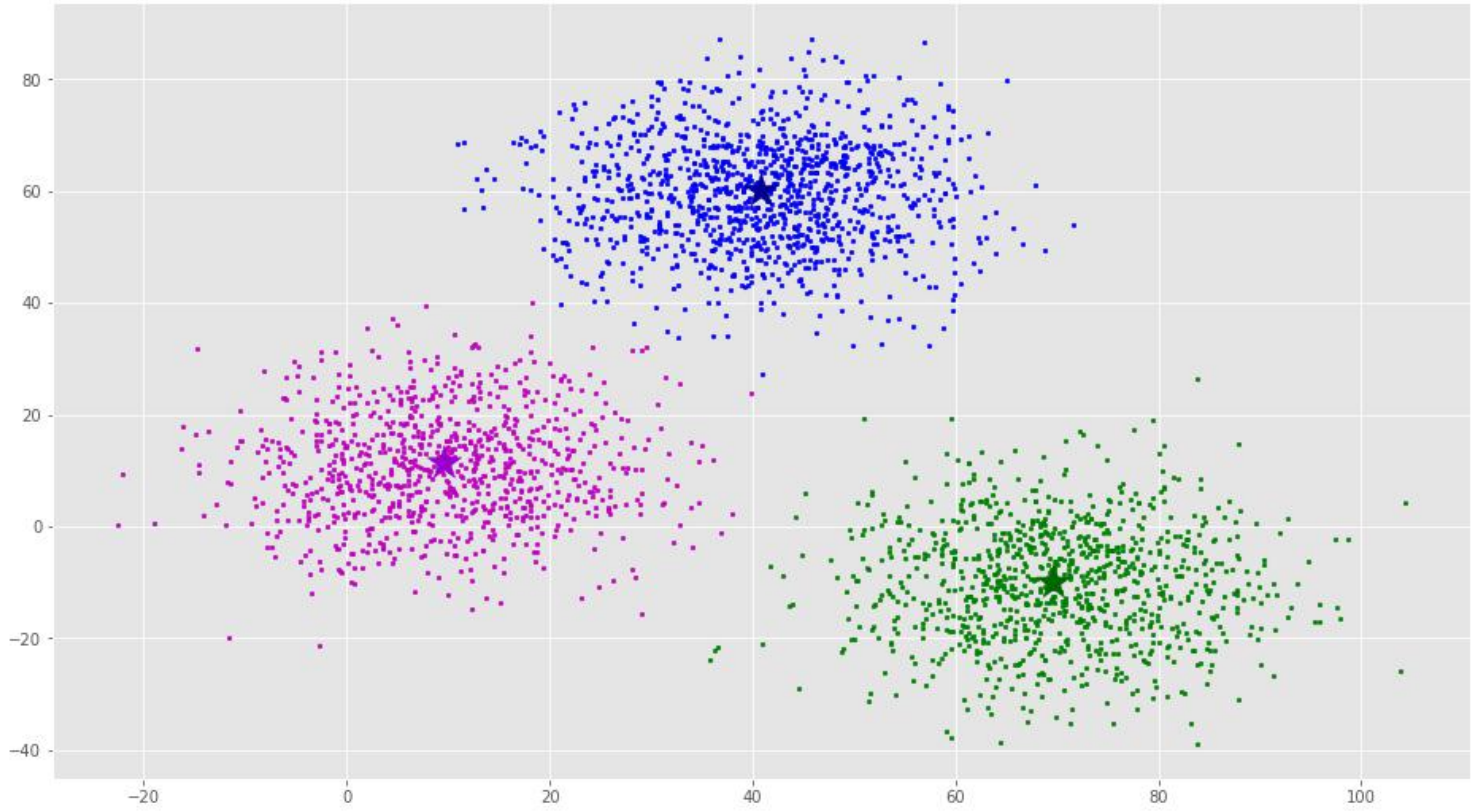
Step 4.A: Assign each data point to its closest centroid



K-Means

How it works

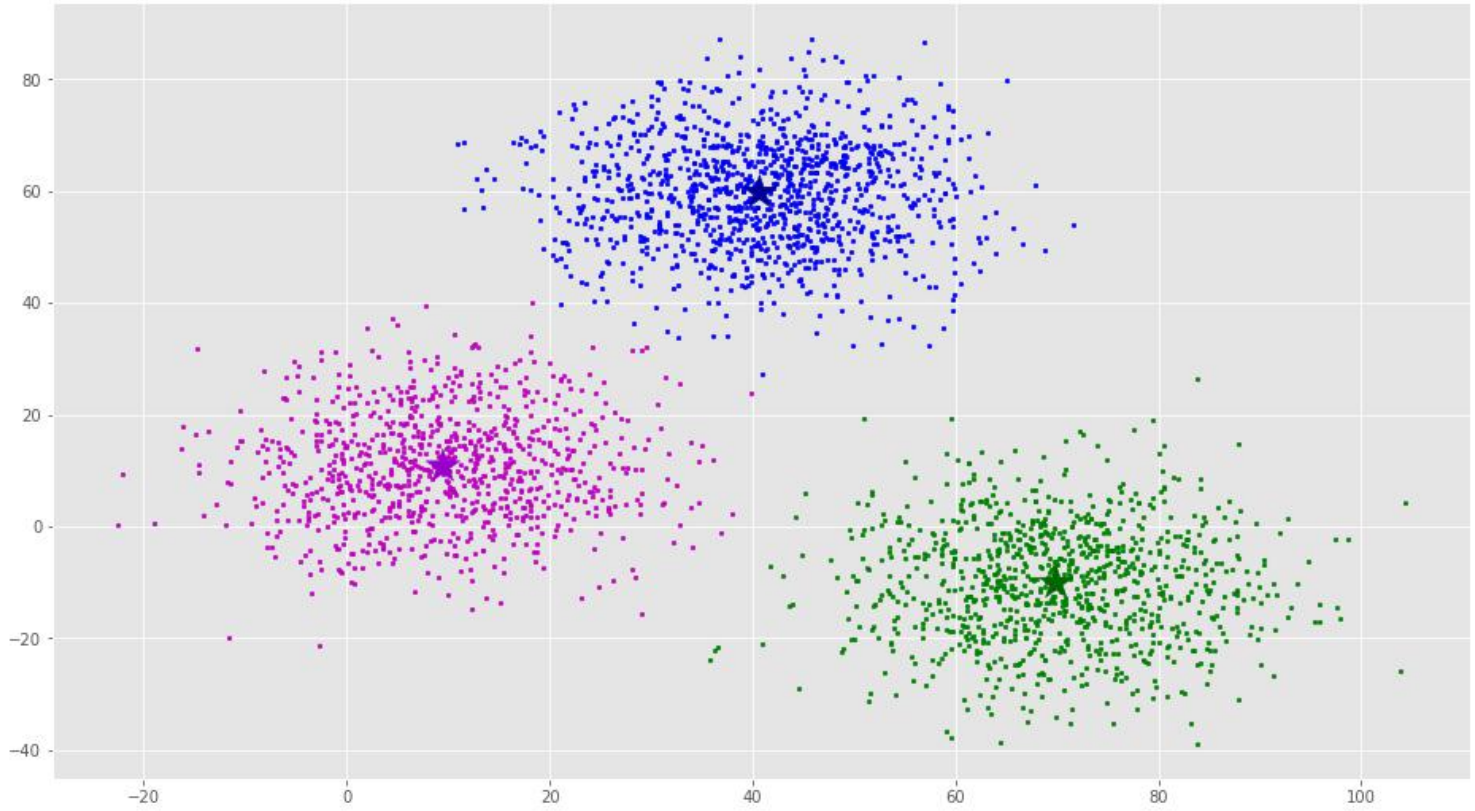
Step 4.B: Update each centroid by computing the **mean** of all points assigned to the centroid



K-Means

How it works

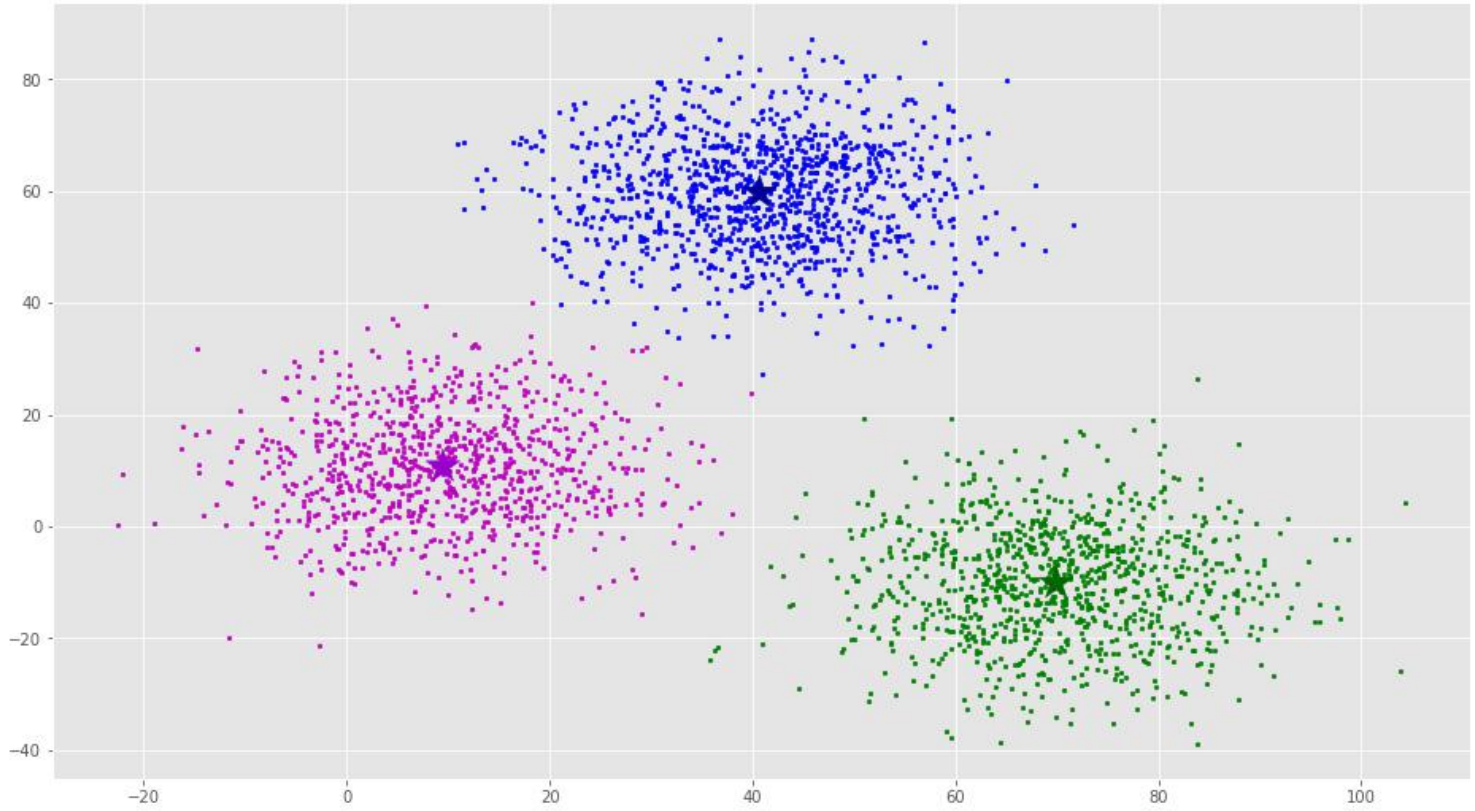
Step 4.B: Update each centroid by computing the **mean** of all points assigned to the centroid



K-Means

How it works

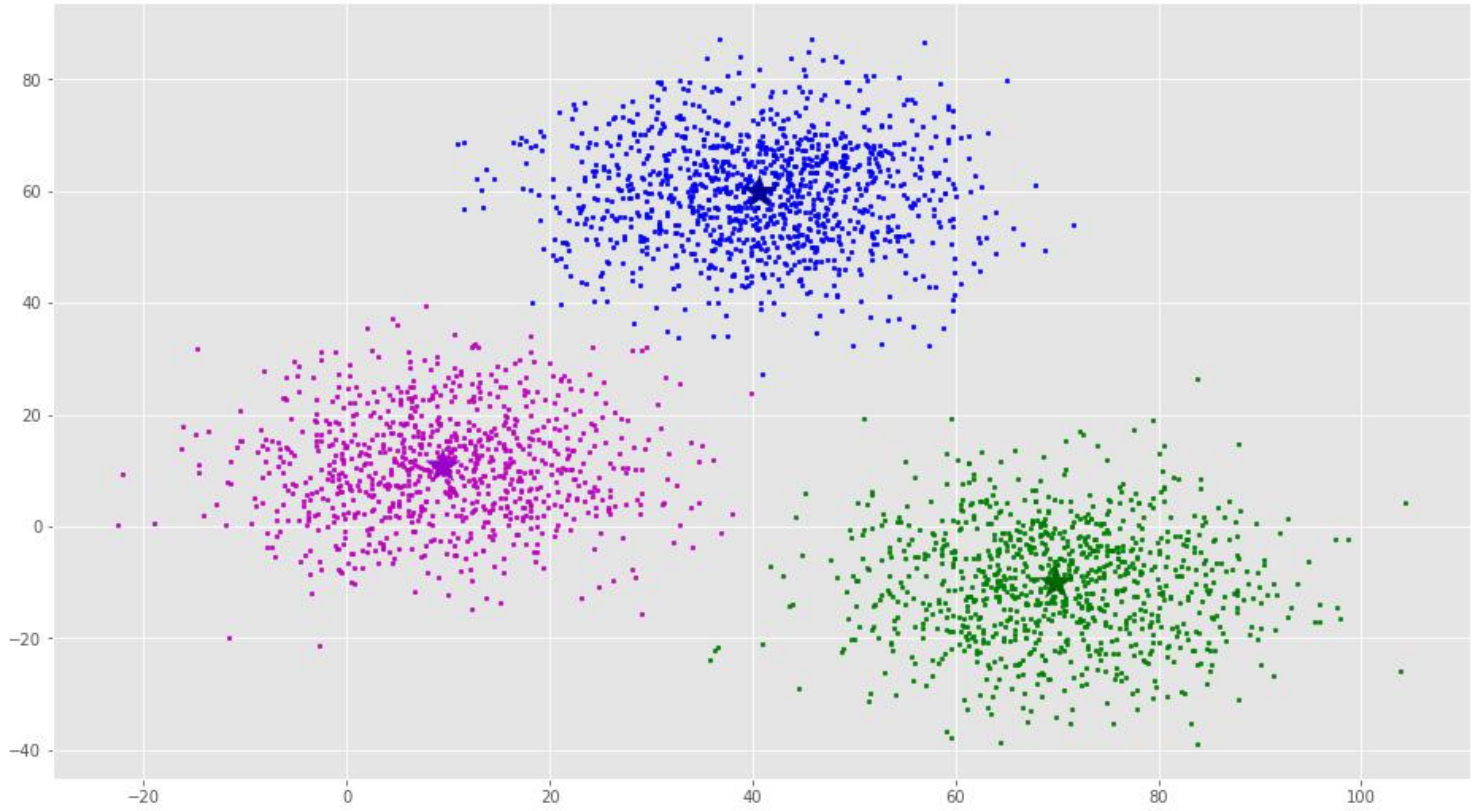
Step 5.A: Assign each data point to its closest centroid



K-Means

How it works

Step 5.A: Assign each data point to its closest centroid

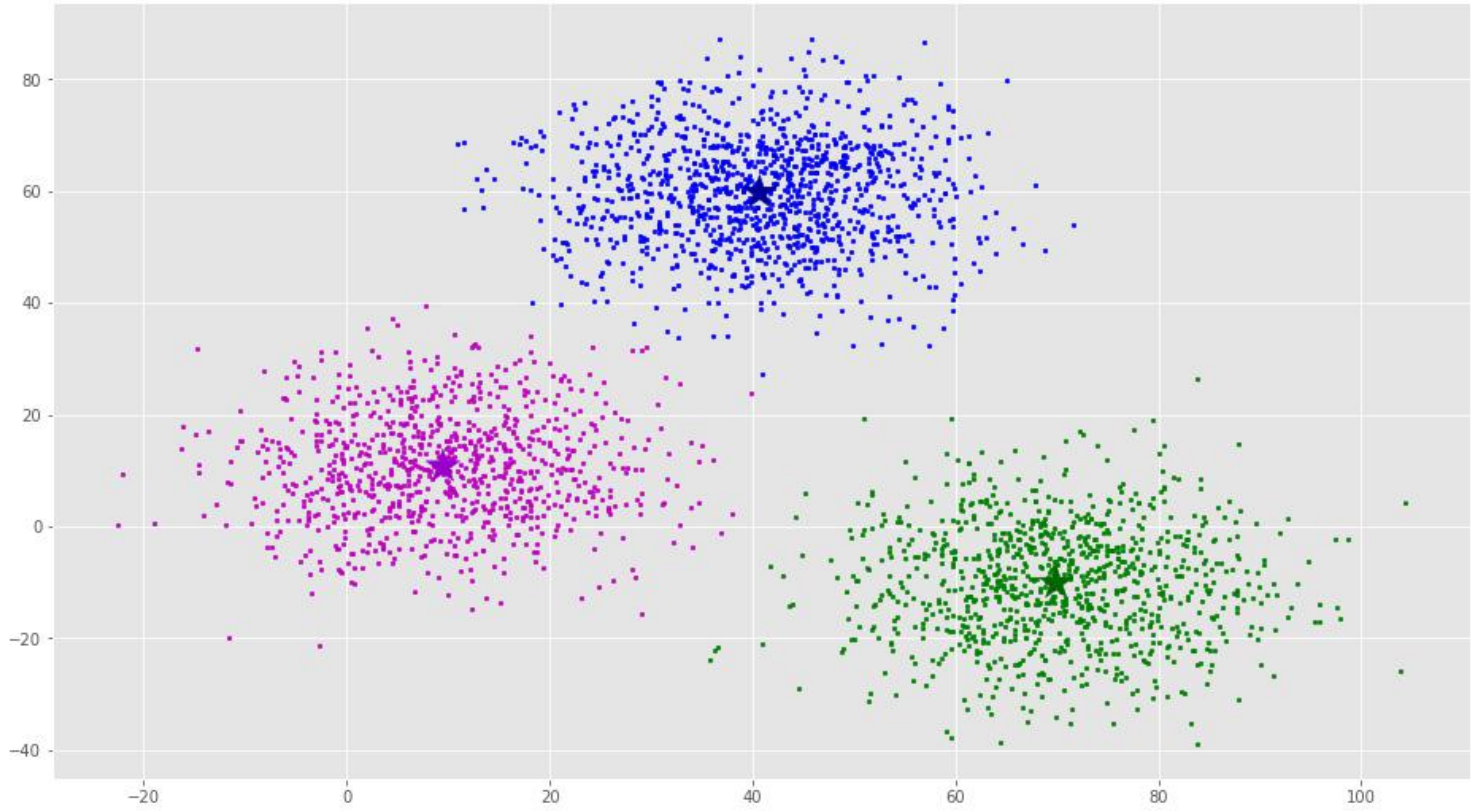


Only one data point changed. Can you spot it?

K-Means

How it works

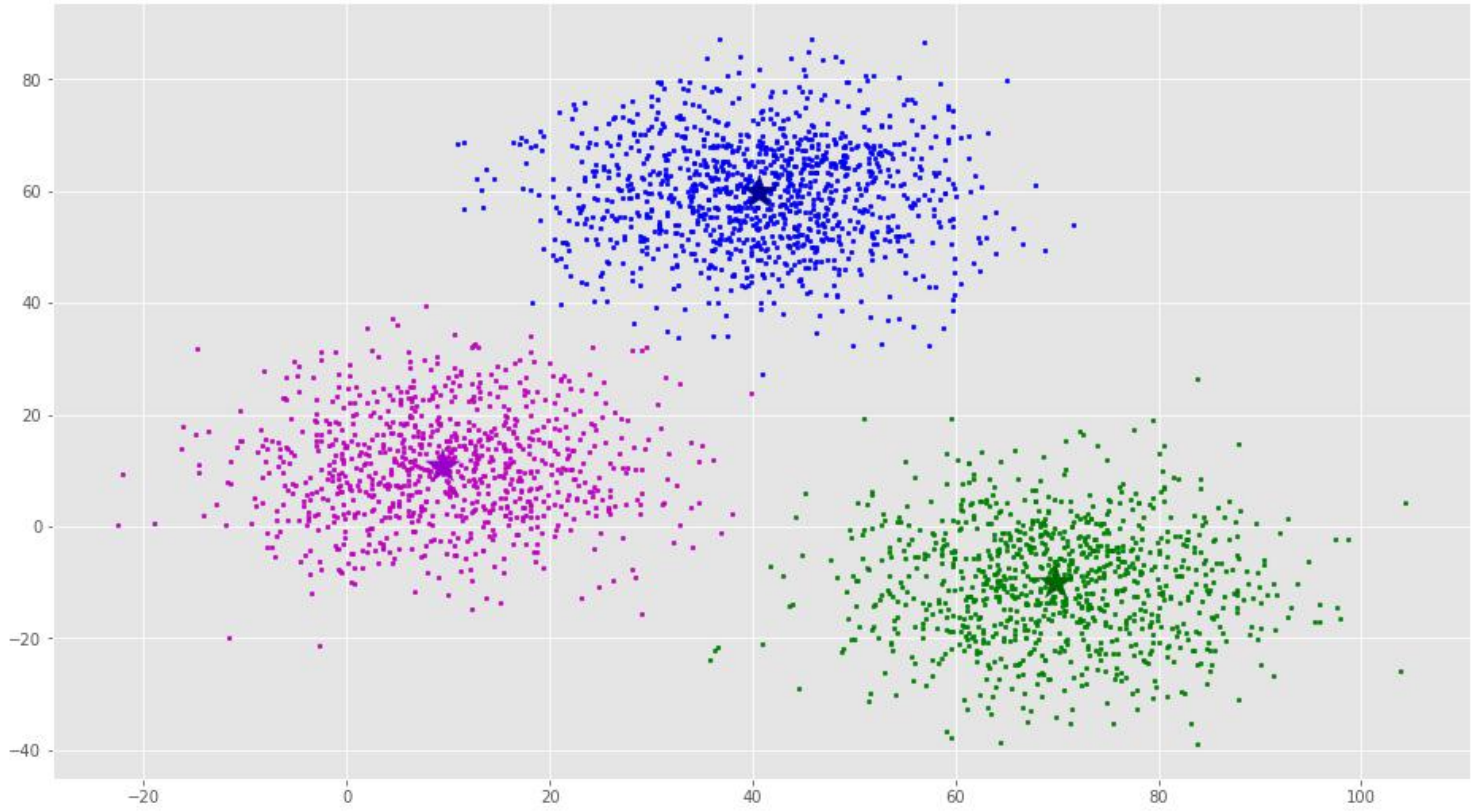
Step 5.B: Update each centroid by computing the **mean** of all points assigned to the centroid



K-Means

How it works

Step 5.B: Update each centroid by computing the **mean** of all points assigned to the centroid



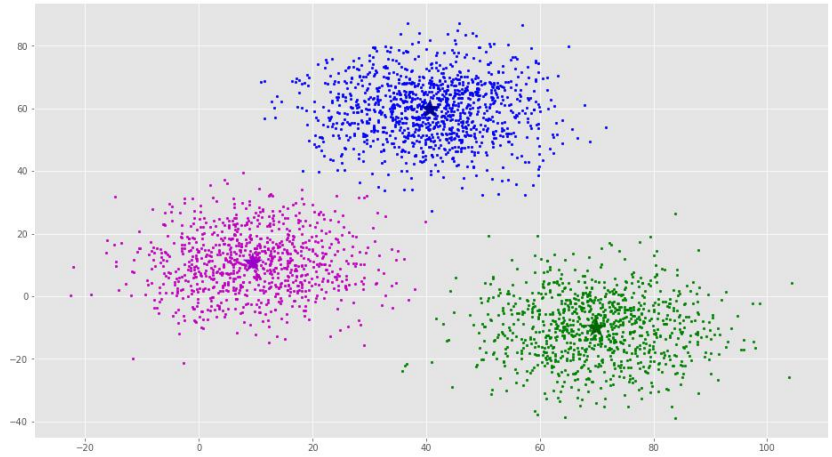
The algorithm has converged.

K-Means

Properties

Pros

- Very fast
- Converges always to a solution
- Easy to understand
- Every point is assigned to a cluster
- Efficient application to new points



K-Means

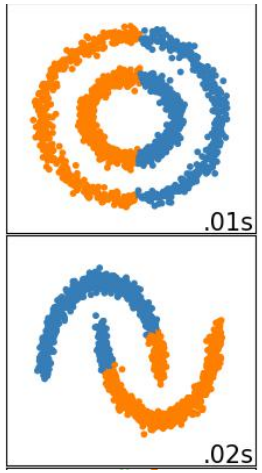
Properties

Pros

- Very fast
- Converges always to a solution
- Easy to understand
- Every point is assigned to a cluster
- Efficient application to new points

Cons

- Always learns convex clusters



K-Means

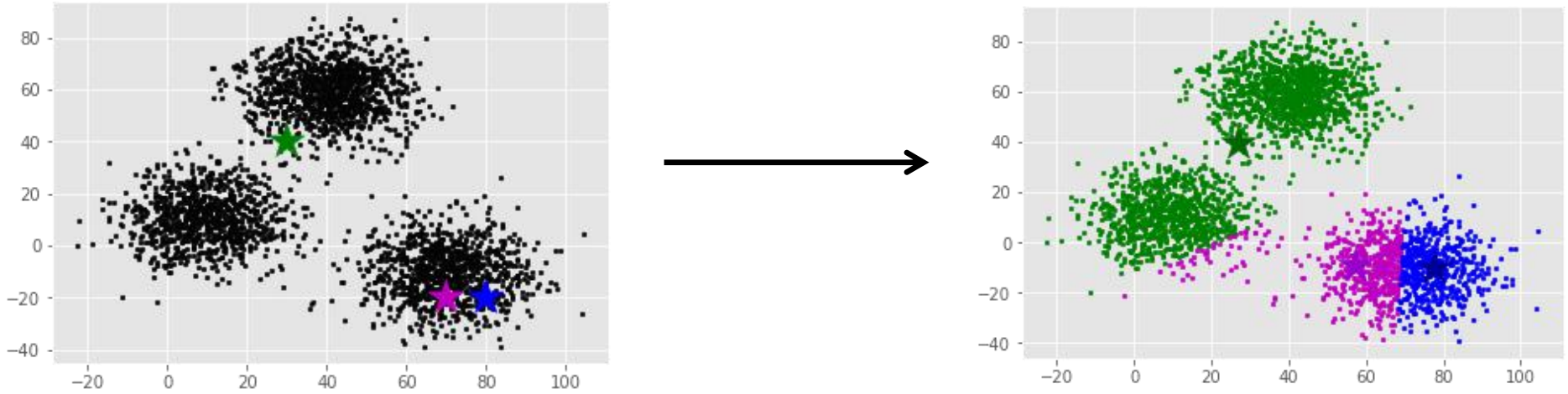
Properties

Pros

- Very fast
- Converges always to a solution
- Easy to understand
- Every point is assigned to a cluster
- Efficient application to new points

Cons

- Always learns convex clusters
- Dependence on initialization



K-Means

Properties

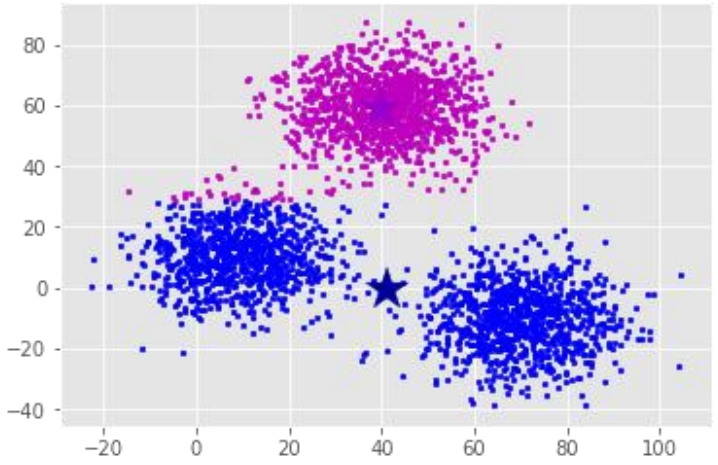
Pros

- Very fast
- Converges always to a solution
- Easy to understand
- Every point is assigned to a cluster
- Efficient application to new points

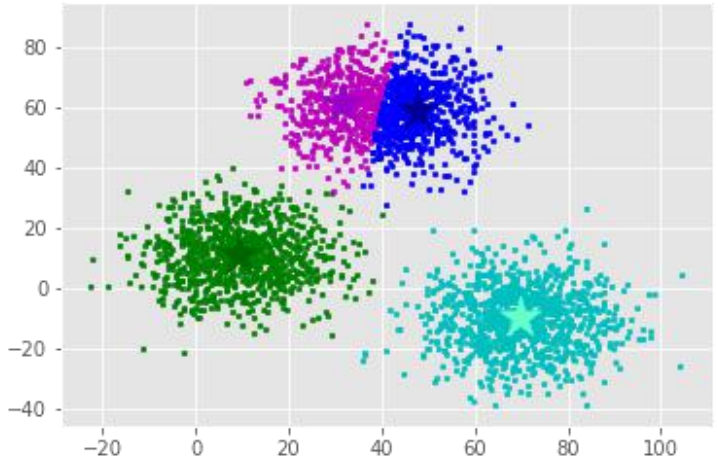
Cons

- Always learns convex clusters
- Dependence on initialization
- Need to know best k (=number of clusters)

k = 2



k = 4



K-Means

Properties

Pros

- Very fast
- Converges always to a solution
- Easy to understand
- Every point is assigned to a cluster
- Efficient application to new points

Cons

- Always learns convex clusters
- Dependence on initialization
- Need to know best k (=number of clusters)
- Sensitive to outliers

k=3

K-Means

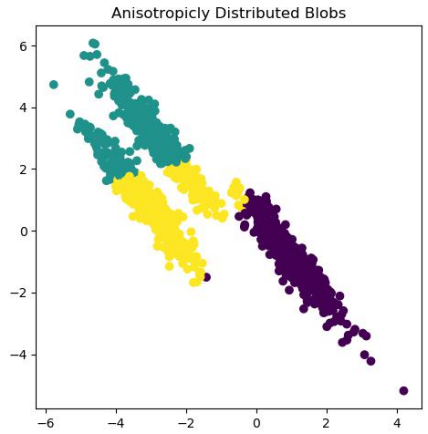
Properties

Pros

- Very fast
- Converges always to a solution
- Easy to understand
- Every point is assigned to a cluster
- Efficient application to new points

Cons

- Always learns convex clusters
- Dependence on initialization
- Need to know best k (=number of clusters)
- Sensitive to outliers
- clusters approximately circular



K-Means

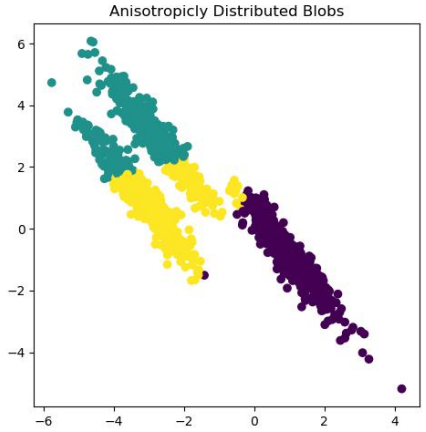
Properties

Pros

- Very fast
- Converges always to a solution
- Easy to understand
- Every point is assigned to a cluster
- Efficient application to new points

Cons

- Always learns convex clusters
- Dependence on initialization
- Need to know best k (=number of clusters)
- Sensitive to outliers
- clusters approximately circular



We will first consider a model that improves on this last issue.

Gaussian Mixture Models

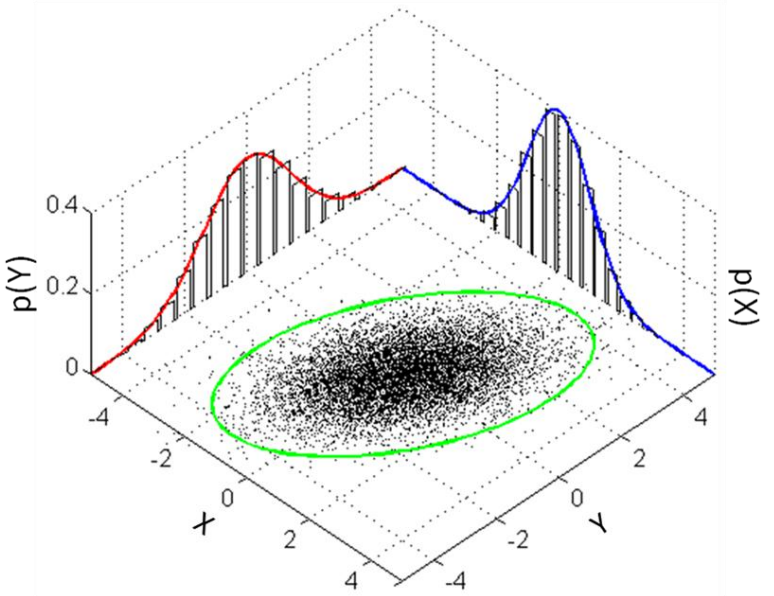
The model

The non-degenerate multivariate Gaussian distribution $\mathcal{N}(\mu, \Sigma)$ in n dimensions is defined by

- its **mean** μ
- and its **covariance matrix** $\sum$

and is the distribution with probability density

$$p(x_1, \ldots, x_n) = \frac{1}{\sqrt{(2\pi)^n \det(\Sigma)}} \exp\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\right)$$



A **Gaussian mixture model** is a distribution $\sum_{i=1}^{k} c_i \cdot \mathcal{N}(\mu_i, \Sigma_i)$

with **weights** $0 \leq c_i \leq 1$ such that $\sum_{i=1}^{k} c_i = 1$

Gaussian Mixture Models

A probabilistic approach to clustering

Idea:

- Describe each **cluster** by a **Gaussian distribution**
- Each cluster is defined by the **mean** and the **covariance** of each Gaussian
- With a **weight** for each cluster, we want to learn a **Gaussian mixture model** that best **fits the data**
- The Gaussian mixture model is learned with an **Expectation-Maximation algorithm** (see next slide)

Parameters: number of clusters **k**

Initializiation: Randomly choose mean, covariance matrix and weight for each cluster :

The EM-algorithm for Gaussian mixture models consists of two steps that are repeated until convergence: The E-step and the M-step.

Gaussian Mixture Models

Expectation Maximization

E-Step:

For each data sample x_α compute its probability to belong to the i-th cluster

$$w_{\alpha,i} = \frac{p_i(x_\alpha) \cdot c_i}{\sum_{j=1}^{k} p_j(x_\alpha) \cdot c_j}$$

M-Step:

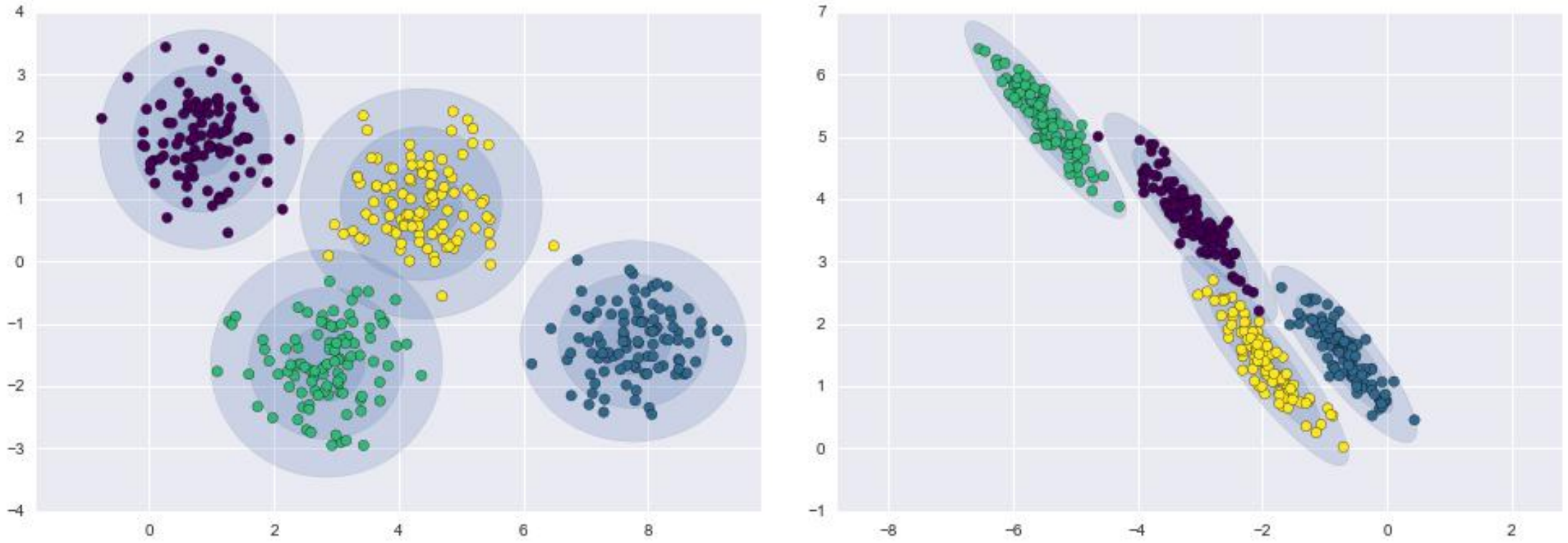
Set $N_i = \sum_\alpha w_{\alpha,i}$ and new weights $c_i = \frac{N_i}{N}$

For each cluster i, find a new mean by $\mu_i = \frac{1}{N_i} \sum_\alpha w_{\alpha,i} x_\alpha$

For each cluster i, find a new covariance matrix $\Sigma_i = \frac{1}{N_i} \sum_\alpha w_{\alpha,i} (x_\alpha - \mu_i)(x_\alpha - \mu_i)^T$

Gaussian Mixture Models

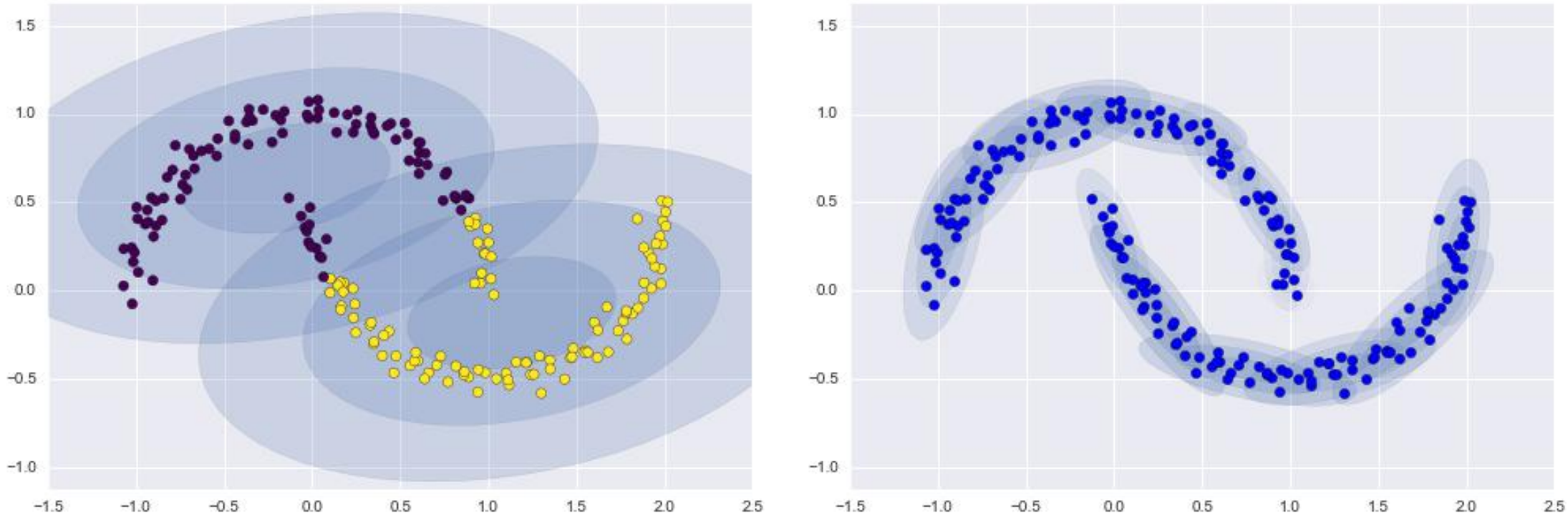
In action



For clusters of elliptoidal shape, Gaussian mixture models do well.

Gaussian Mixture Models

In action

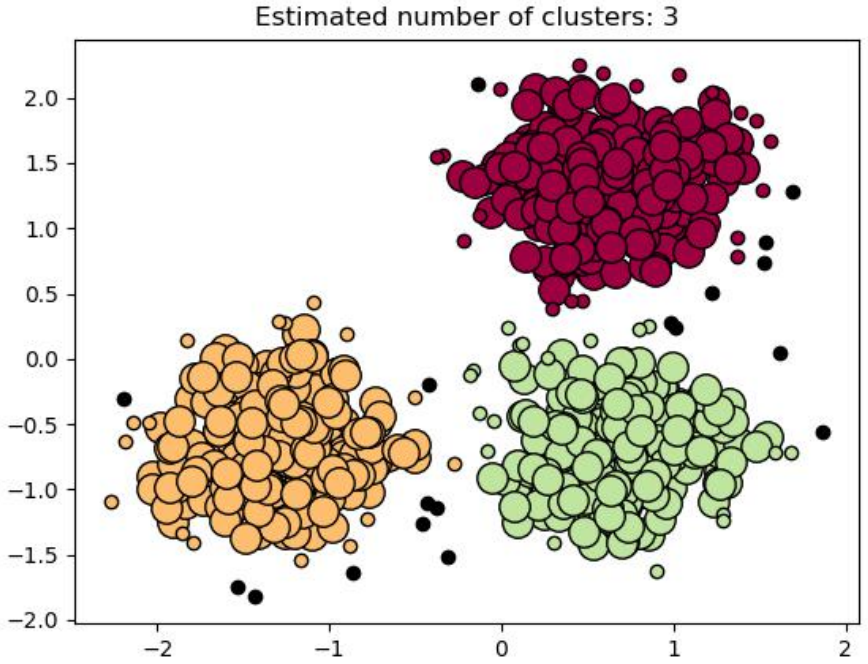


For non-elliptoidal clusters, we need to decompose cluster into many small subclusters

an alternative to K-Means and GMMs

We would like an option to K-Means and GMMs that....

- ... is resistant to outliers
- ... can find clusters of arbitrary size and form
- ... is independent of initialization
- ... is also quite fast
- ... eventually ends with clustered dataset
- ... can be efficiently applied to new data points
- ... can automatically find a suitable number of clusters

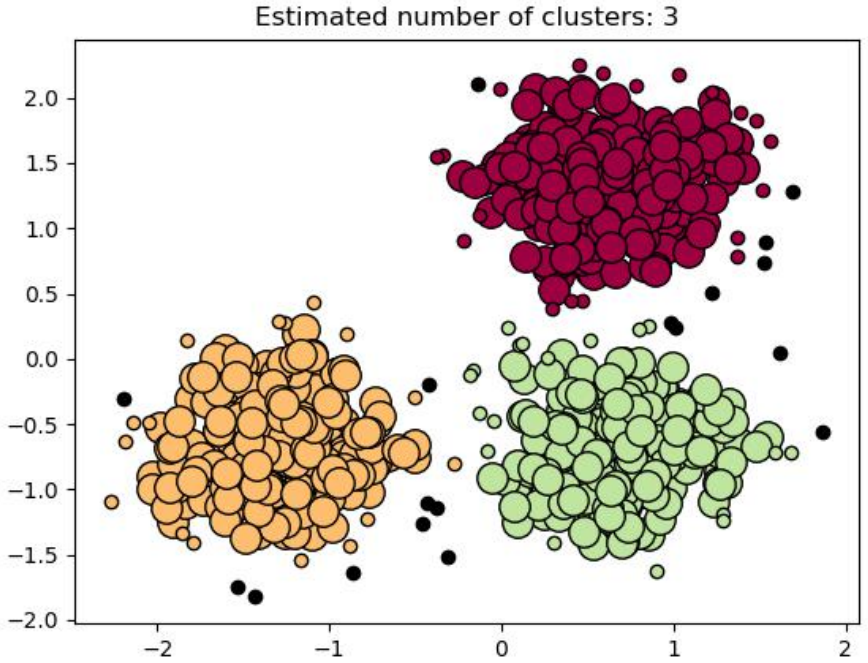


DBSCAN

as an alternative to K-Means and GMMs

We would like an option to K-Means and GMMs that....

- ... is resistant to outliers
- ... can find clusters of arbitrary size and form
- ... is independent of initialization
- ... is also quite fast
- ... eventually ends with clustered dataset
- ... can be efficiently applied to new data points
- ... can automatically find a suitable number of clusters



DBSCAN satisfies almost all of these properties

- „DB“ stands for **density based**
- **clusters are dense regions** of points
- points that are far from all other points get their own cluster, i.e., are marked as outliers

DBSCAN

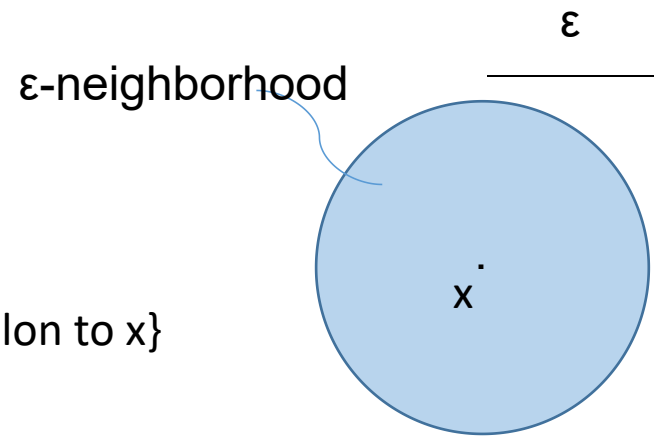
Parameters and Terminology

DBScan depends on the choice of two **parameters**:

- **Neighborhood distance ε**
- **minimal number of neighbors in a cluster m**

Terminology of DBSCAN

- **ε-neighborhood** of x = { all points z that have a distance less than epsilon to x}



DBSCAN

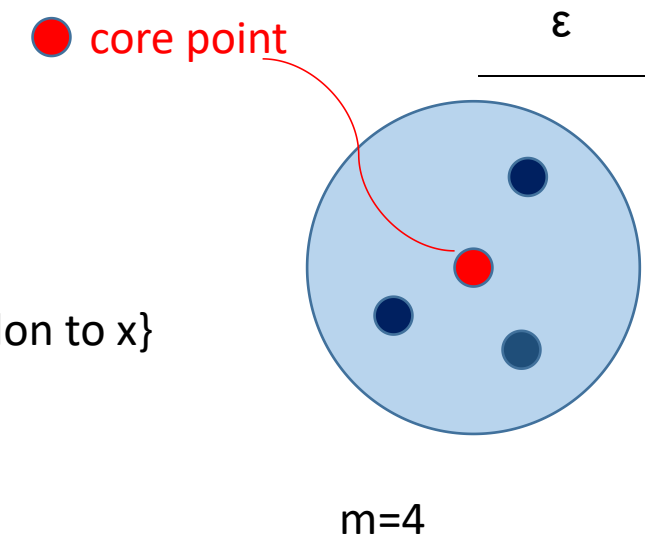
Parameters and Terminology

DBScan depends on the choice of two **parameters**:

- **Neighborhood distance ε**
- **minimal number of neighbors in a cluster m**

Terminology of DBSCAN

- **ε-neighborhood** of x = { all points z that have a distance less than epsilon to x}
- a **core point** has at least m-1 other points in its epsilon-neighborhood

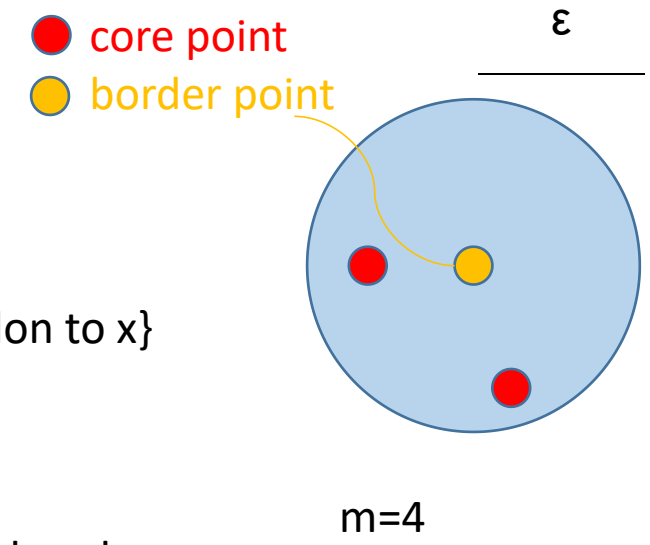


DBSCAN

Parameters and Terminology

DBScan depends on the choice of two **parameters**:

- **Neighborhood distance ε**
- **minimal number of neighbors in a cluster m**



Terminology of DBSCAN

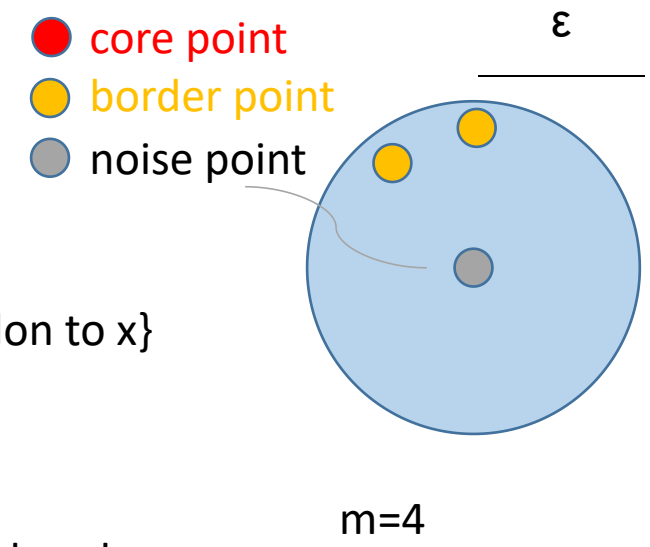
- **ε-neighborhood** of x = { all points z that have a distance less than epsilon to x}
- a **core point** has at least m-1 other points in its epsilon-neighborhood
- a **border point** is not a core point but has a core point in its ε-neighborhood

DBSCAN

Parameters and Terminology

DBScan depends on the choice of two **parameters**:

- **Neighborhood distance ε**
- **minimal number of neighbors in a cluster m**



Terminology of DBSCAN

- **ε-neighborhood** of x = { all points z that have a distance less than epsilon to x}
- a **core point** has at least m-1 other points in its epsilon-neighborhood
- a **border point** is not a core point but has a core point in its ε-neighborhood
- a **noise point** is not a core point and has no core point in its ε-neighborhood

DBSCAN

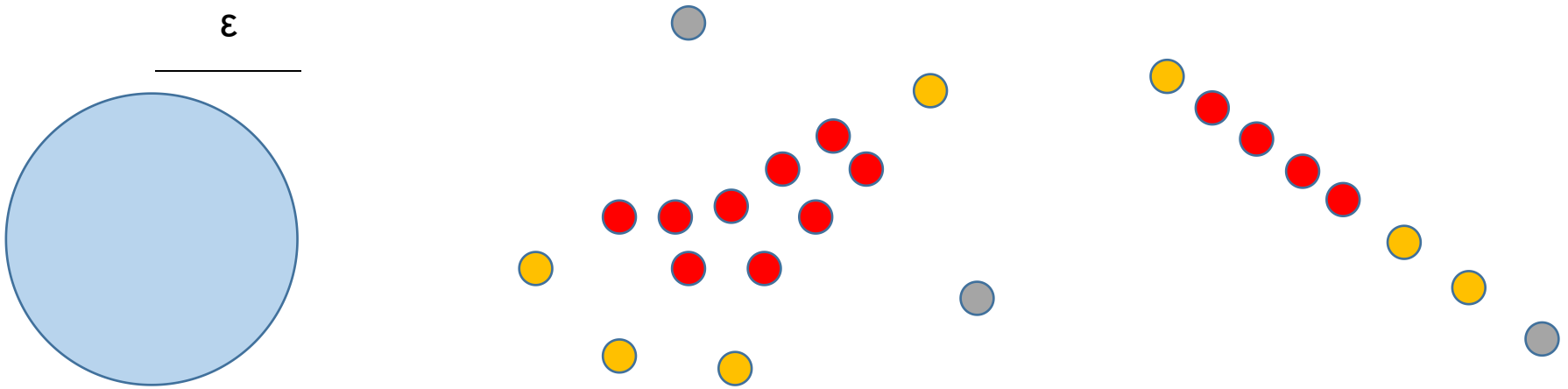
Parameters and Terminology

Terminology of DBSCAN

- **ε-neighborhood** of x = { all points z that have a distance less than epsilon to x}
- a **core point** has at least m-1 other points in its epsilon-neighborhood
- a **border point** is not a core point but has a core point in its ε-neighborhood
- a **noise point** is not a core point and has no core point in its ε-neighborhood

core point
border point
noise point

m=4



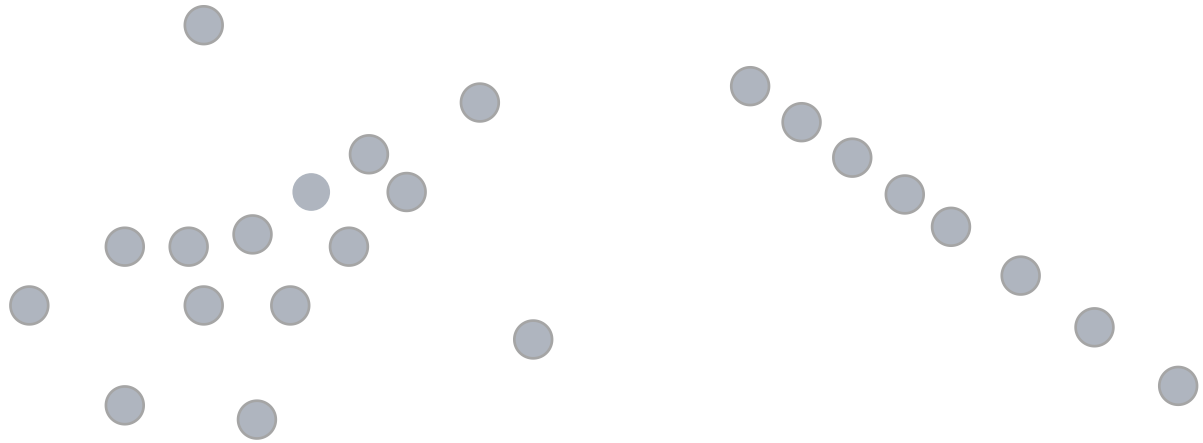
DBSCAN

How it works

- **ε-neighborhood** of x = { all points z that have a distance less than epsilon to x}
- a **core point** has at least m-1 other points in its epsilon-neighborhood
- a **border point** is not a core point but has a core point in its ε-neighborhood
- a **noise point** is not a core point and has no core point in its ε-neighborhood

core point
border point
noise point

1.) Choose a point randomly



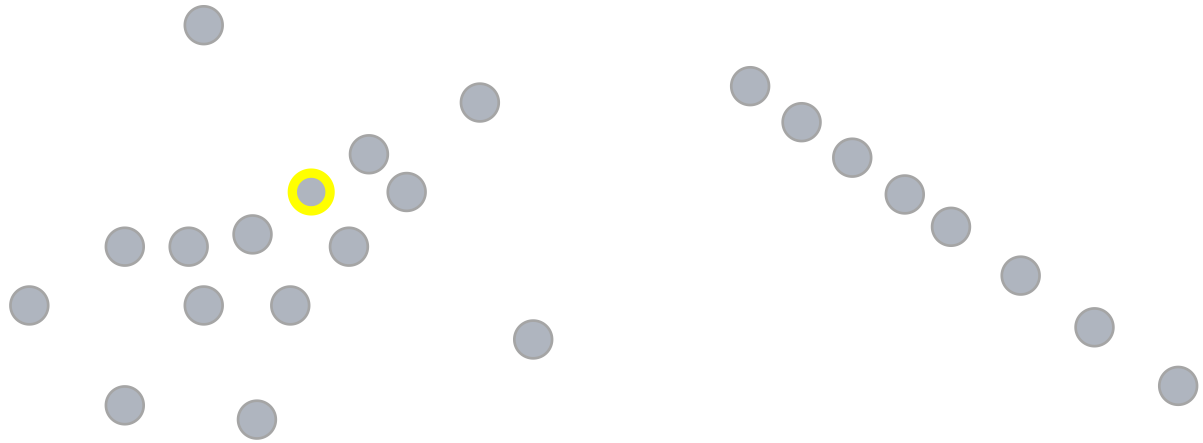
DBSCAN

How it works

- **ε-neighborhood** of x = { all points z that have a distance less than epsilon to x}
- a **core point** has at least m-1 other points in its epsilon-neighborhood
- a **border point** is not a core point but has a core point in its ε-neighborhood
- a **noise point** is not a core point and has no core point in its ε-neighborhood

core point
border point
noise point

1.) Choose a point randomly



DBSCAN

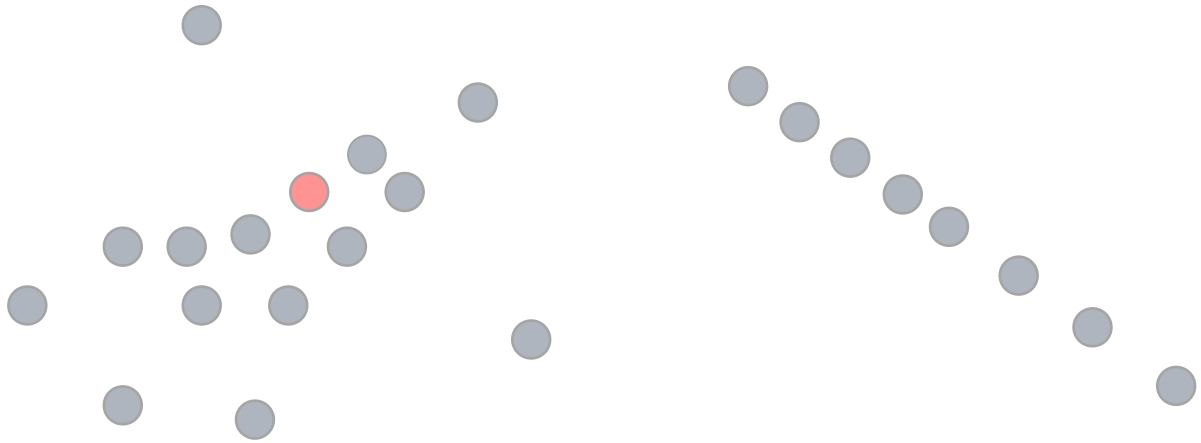
How it works

- **ε-neighborhood** of x = { all points z that have a distance less than epsilon to x}
- a **core point** has at least m-1 other points in its epsilon-neighborhood
- a **border point** is not a core point but has a core point in its ε-neighborhood
- a **noise point** is not a core point and has no core point in its ε-neighborhood

core point
border point
noise point

1.) Choose a point randomly

2.) Iterate until all nodes labeled:
For the current point:
If there are at least m-1 points in its neighborhood,
then label it as core point and choose neighbor
Else:
Label as noise or border



DBSCAN

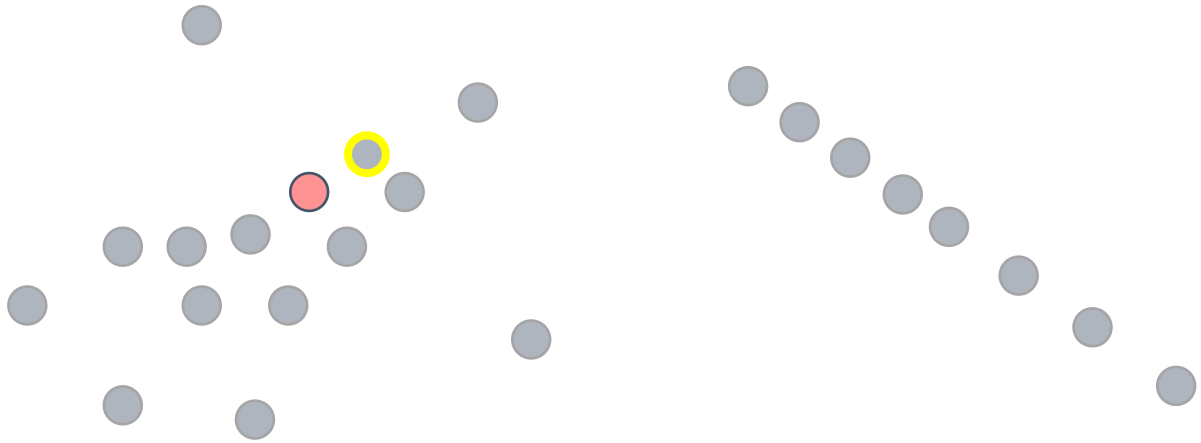
How it works

- **ε-neighborhood** of x = { all points z that have a distance less than epsilon to x}
- a **core point** has at least m-1 other points in its epsilon-neighborhood
- a **border point** is not a core point but has a core point in its ε-neighborhood
- a **noise point** is not a core point and has no core point in its ε-neighborhood

core point
border point
noise point

1.) Choose a point randomly

2.) Iterate until all nodes labeled:
For the current point:
If there are at least m-1 points in its neighborhood,
then label it as core point and choose neighbor
Else:
Label as noise or border



DBSCAN

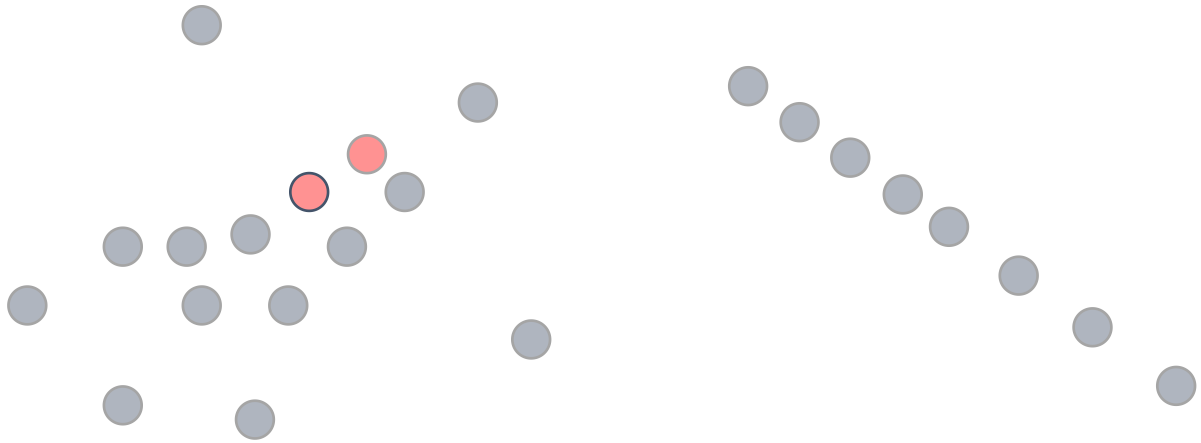
How it works

- **ε-neighborhood** of x = { all points z that have a distance less than epsilon to x}
- a **core point** has at least m-1 other points in its epsilon-neighborhood
- a **border point** is not a core point but has a core point in its ε-neighborhood
- a **noise point** is not a core point and has no core point in its ε-neighborhood

core point
border point
noise point

1.) Choose a point randomly

2.) Iterate until all nodes labeled:
For the current point:
If there are at least m-1 points in its neighborhood,
then label it as core point and choose neighbor
Else:
Label as noise or border



DBSCAN

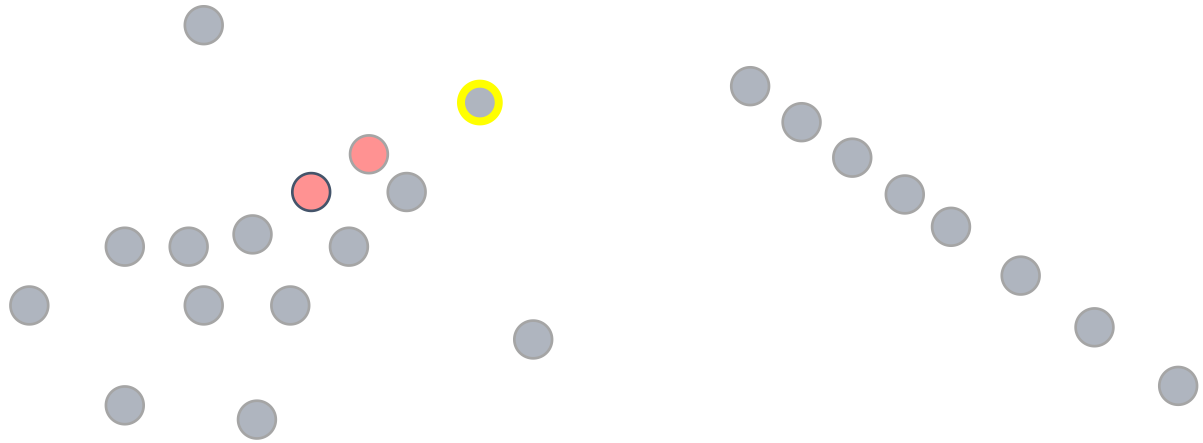
How it works

- **ε-neighborhood** of x = { all points z that have a distance less than epsilon to x}
- a **core point** has at least m-1 other points in its epsilon-neighborhood
- a **border point** is not a core point but has a core point in its ε-neighborhood
- a **noise point** is not a core point and has no core point in its ε-neighborhood

core point
border point
noise point

1.) Choose a point randomly

2.) Iterate until all nodes labeled:
For the current point:
If there are at least m-1 points in its neighborhood,
then label it as core point and choose neighbor
Else:
Label as noise or border



DBSCAN

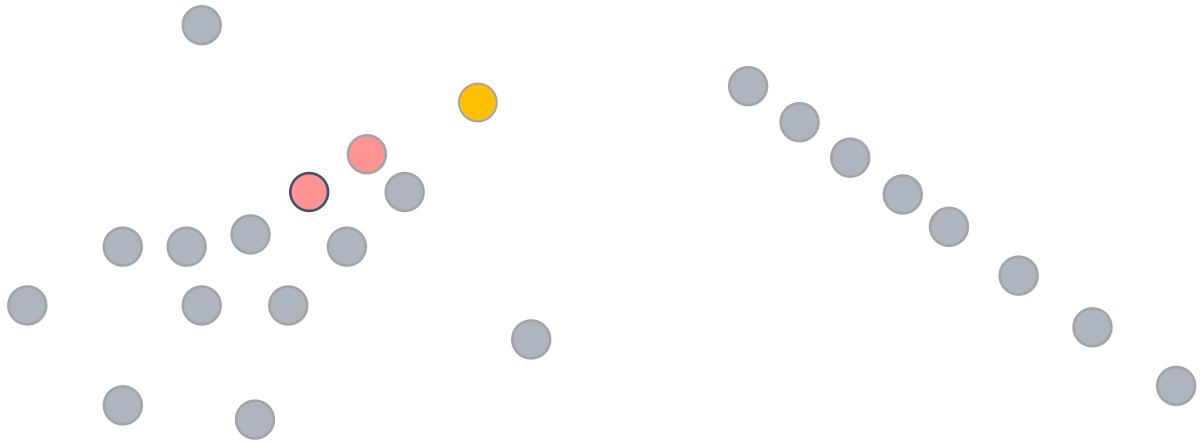
How it works

- **ε-neighborhood** of x = { all points z that have a distance less than epsilon to x}
- a **core point** has at least m-1 other points in its epsilon-neighborhood
- a **border point** is not a core point but has a core point in its ε-neighborhood
- a **noise point** is not a core point and has no core point in its ε-neighborhood

core point
border point
noise point

1.) Choose a point randomly

2.) Iterate until all nodes labeled:
For the current point:
If there are at least m-1 points in its neighborhood,
then label it as core point and choose neighbor
Else:
Label as noise or border



DBSCAN

Properties

Pros

- clusters of arbitrary size and form
- automatically finds suitable number of clusters
- Can be used for outlier detection
- core points independent of initialization

DBSCAN

Properties

Pros

- clusters of arbitrary size and form
- automatically finds suitable number of clusters
- Can be used for outlier detection
- core points independent of initialization

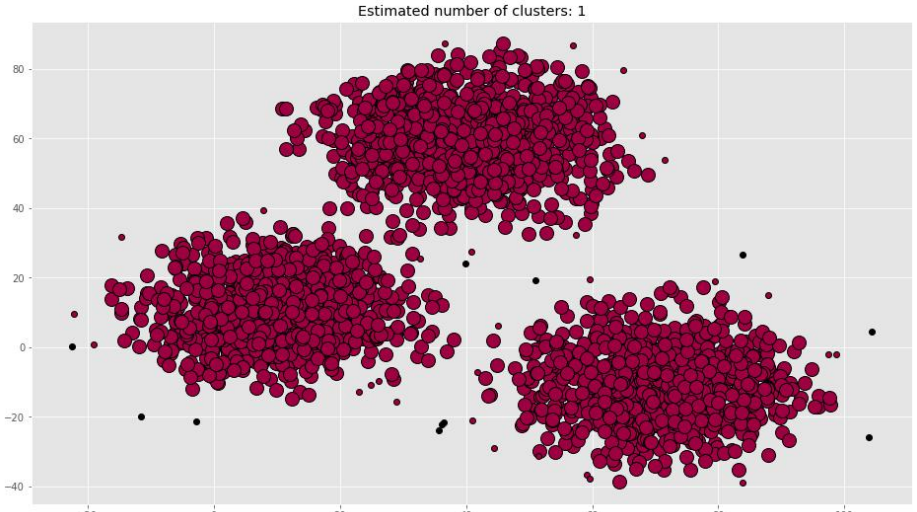
Cons

- All clusters same density
- A good neighborhood distance ε is hard to find
- Need to know best m (=min size of clusters)
- Assignment of border and noise points depends on random order in algorithm

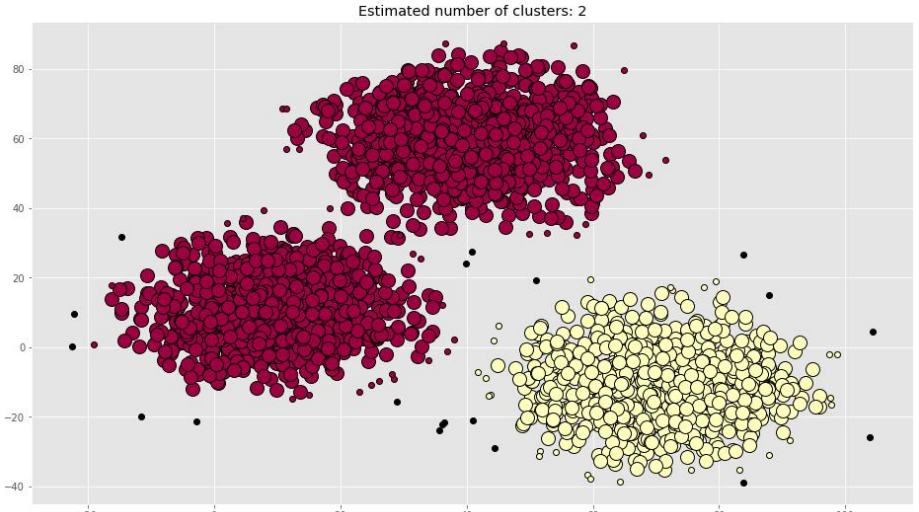
DBSCAN

Examples

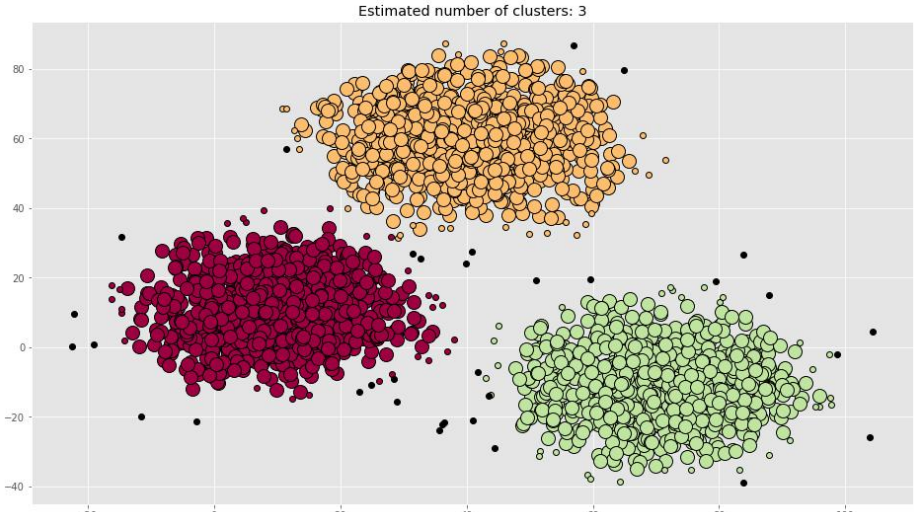
eps=8
m=10



eps=7
m=10



eps=6
m=10



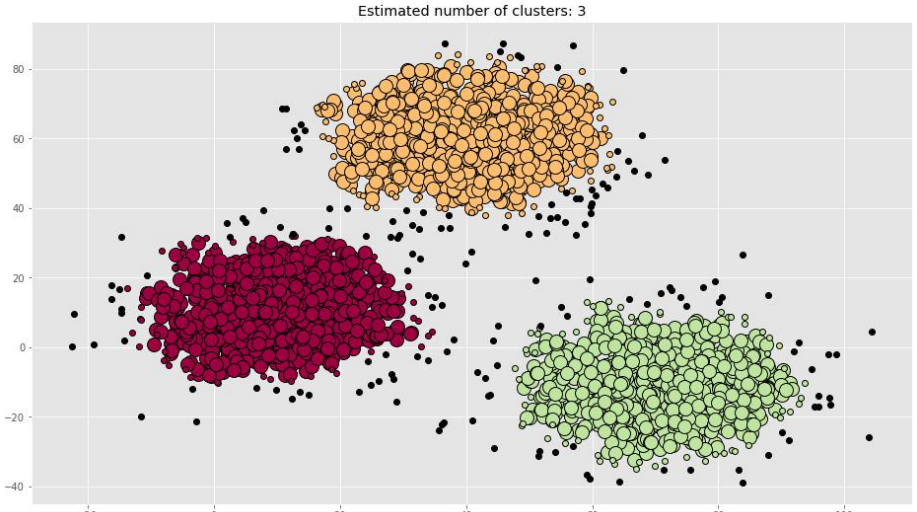
With too large epsilon, can not separate clusters

Note: Colors here denote cluster assignment, NOT type of node (core vs boundary)!
Outliers are black.

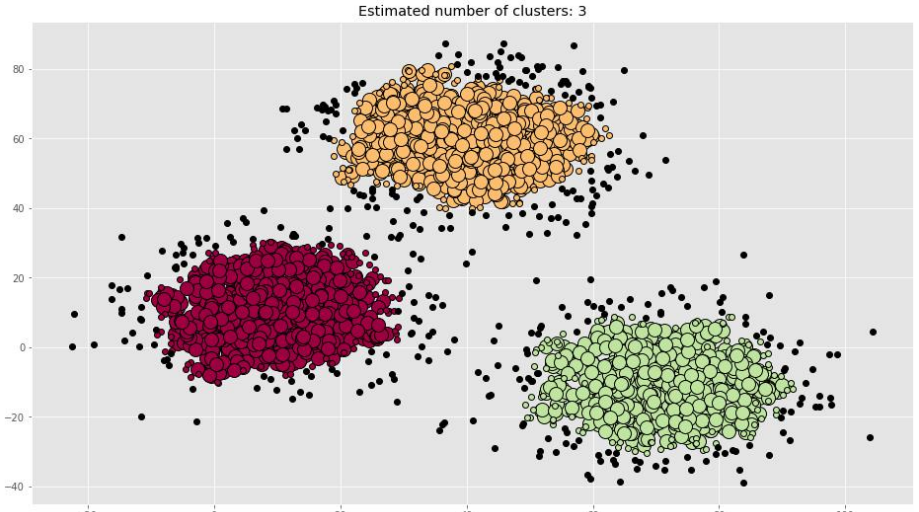
DBSCAN

Examples

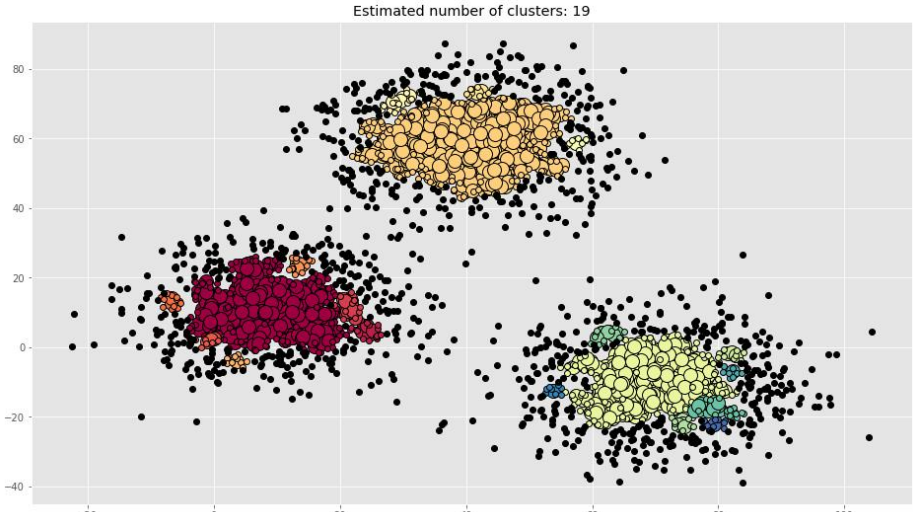
eps=4
m=10



eps=3
m=10



eps=2
m=10



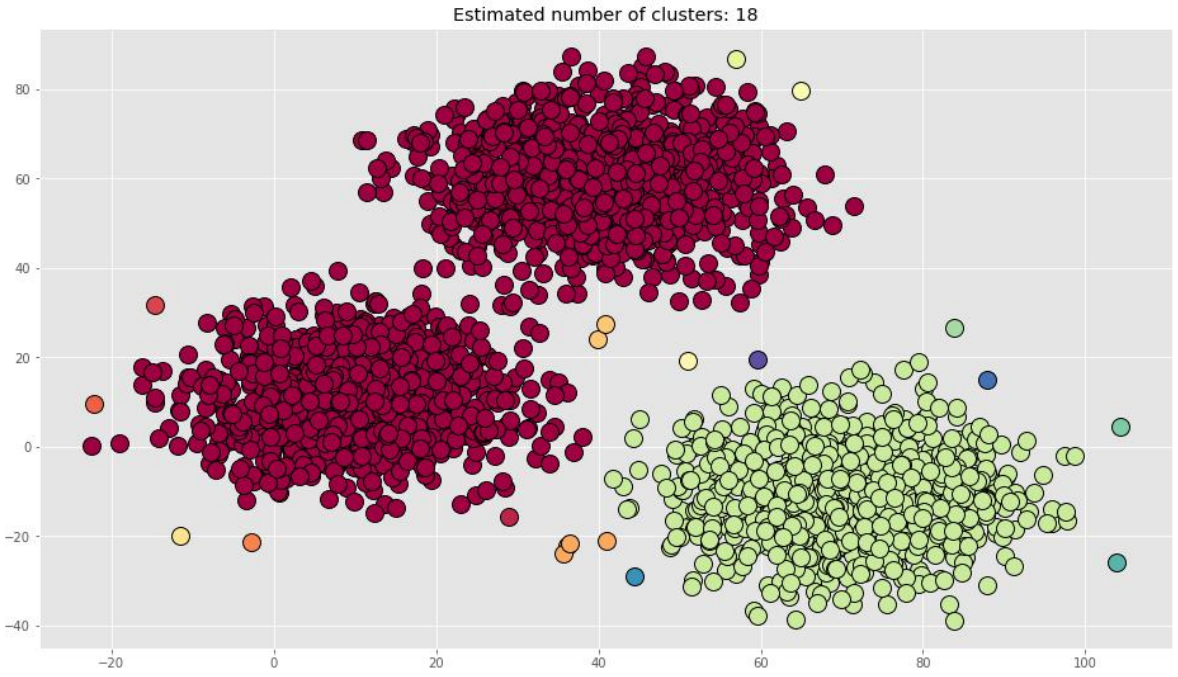
With too small epsilon, get many small clusters and many outliers

Note: Colors here denote cluster assignment, NOT type of node (core vs boundary)!
Outliers are black.

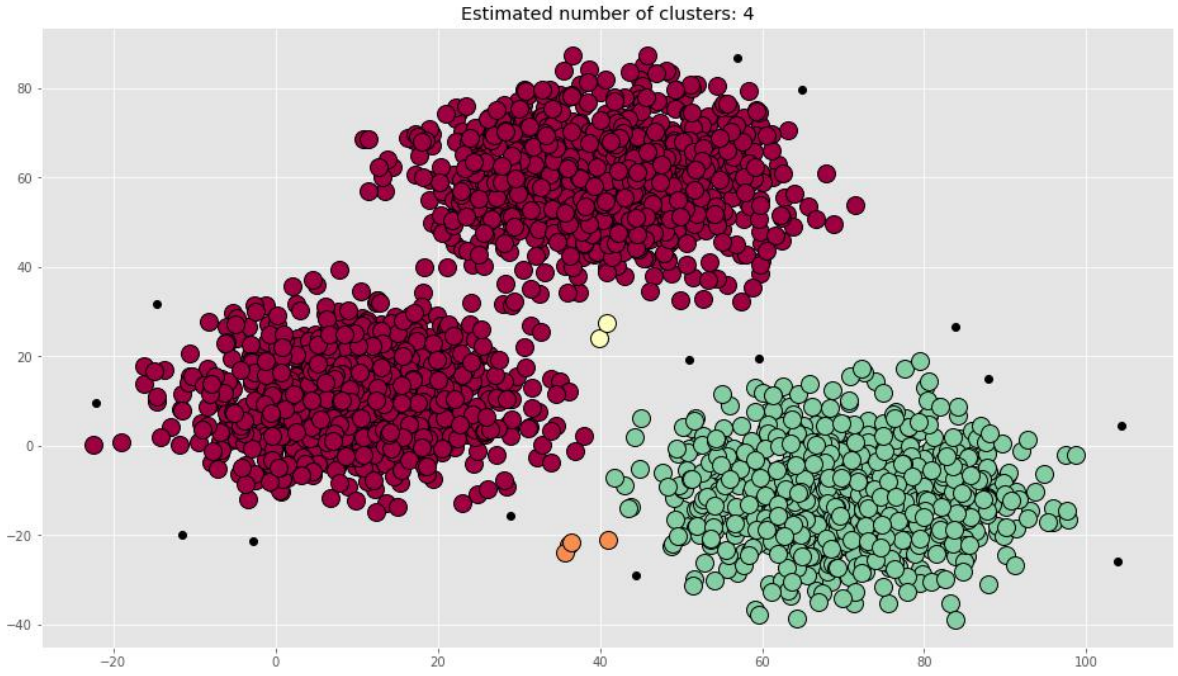
DBSCAN

Examples

eps=6
m=1



eps=6
m=2

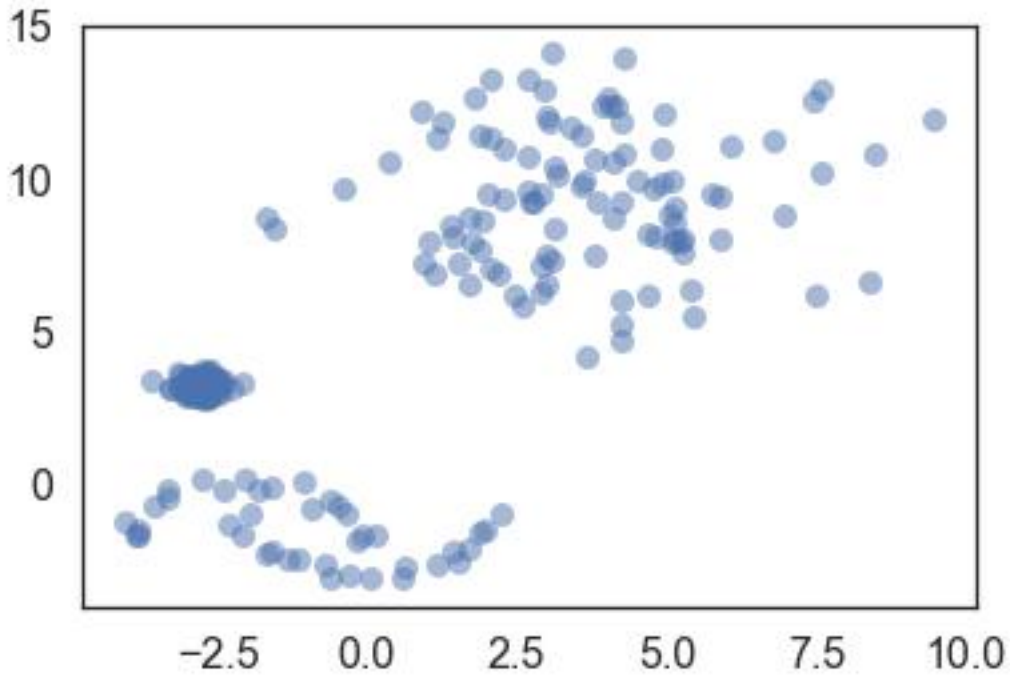


DBSCAN

Examples

For DBSCAN we fix the neigborhood parameter, which defines the density of points in a cluster.

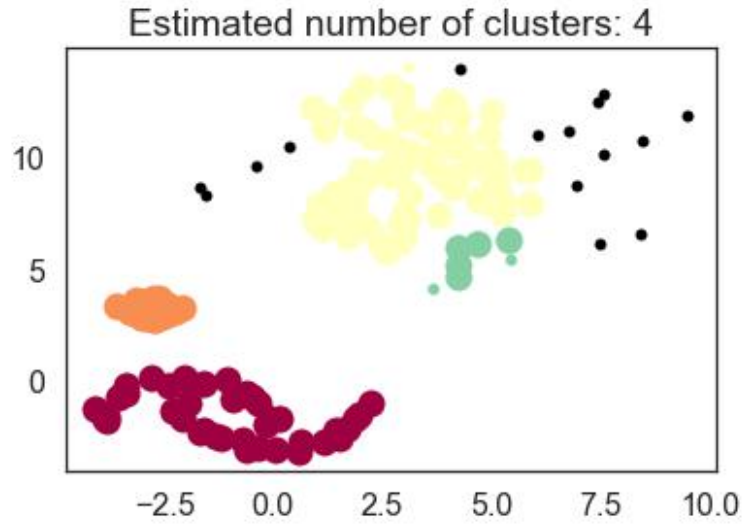
What if clusters have different densities?



DBSCAN

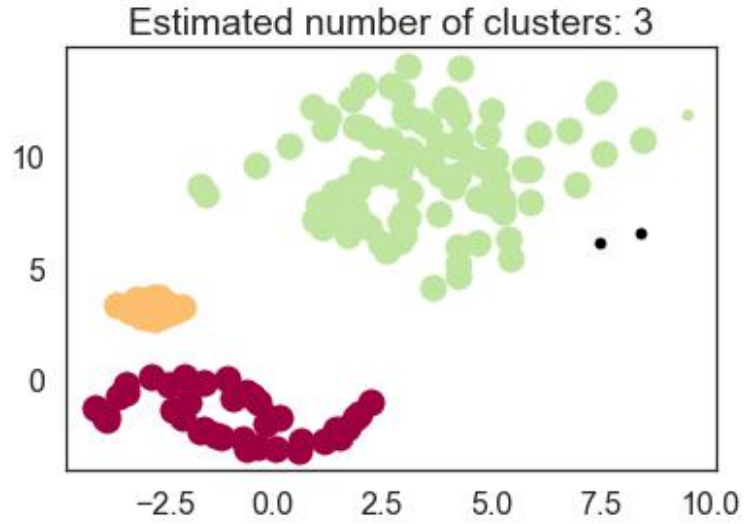
Examples

eps=2

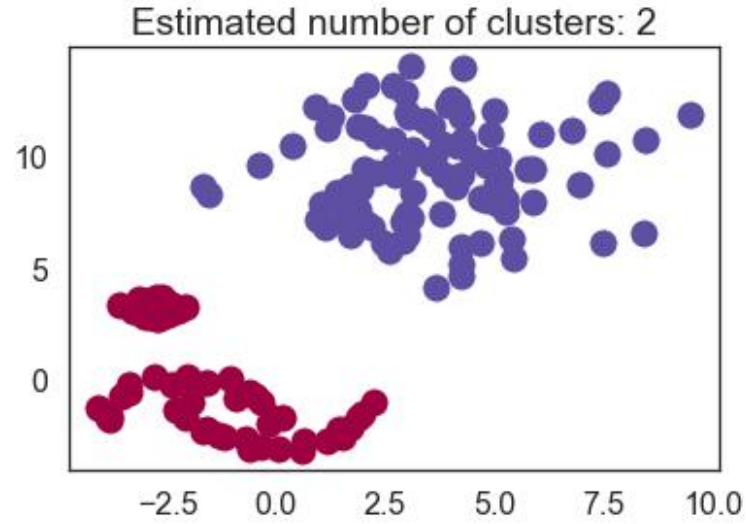


There is no good common neighborhood parameter. Either we get outliers that should belong to a cluster, or we combine clusters that should be separated.

eps=3



eps=4

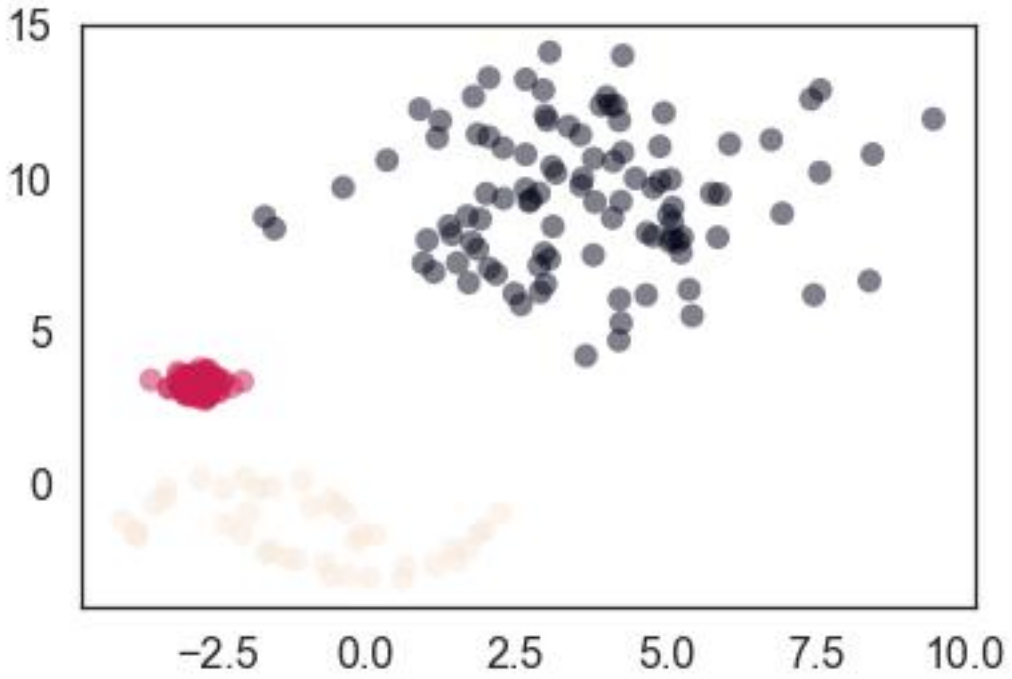


HDBSCAN

Properties

HDBSCAN (Hierarchical DBSCAN)...

- ... allows **different densities** of clusters
- ... is very efficient algorithm (faster than DBSCAN)
- ... requires only one parameter: Minimum cluster size



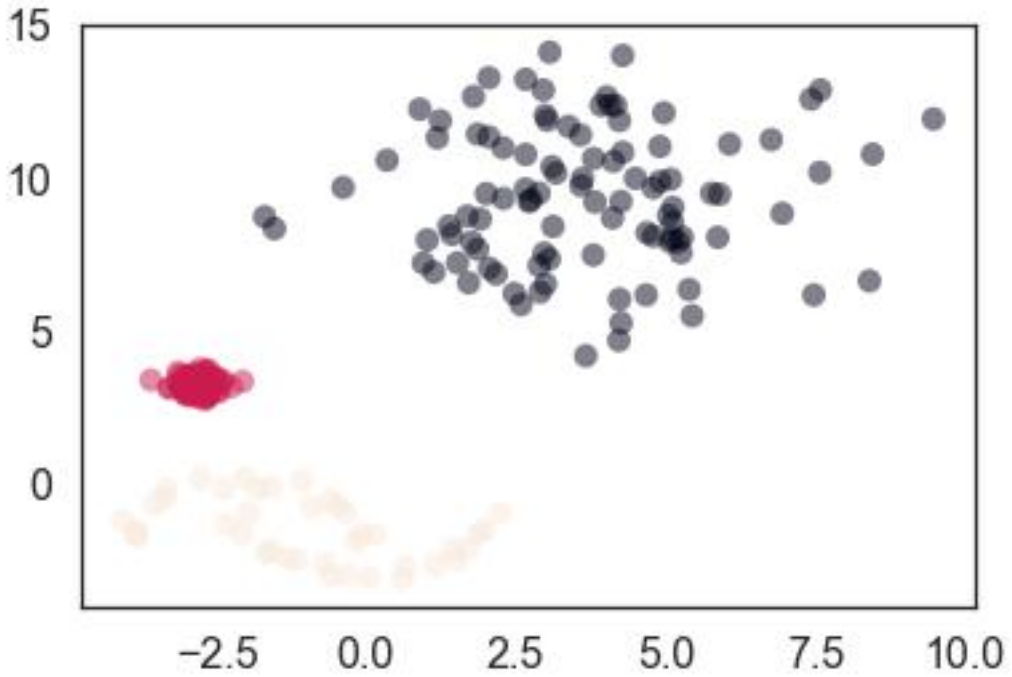
HDBSCAN

Properties

HDBSCAN (Hierarchical DBSCAN)...

- ... allows **different densities** of clusters
- ... is very efficient algorithm
- ... requires only one parameter: Minimum cluster size

Before going through HDBSCAN we will learn about a whole class of clustering methods:
--> Hierarchical Clustering

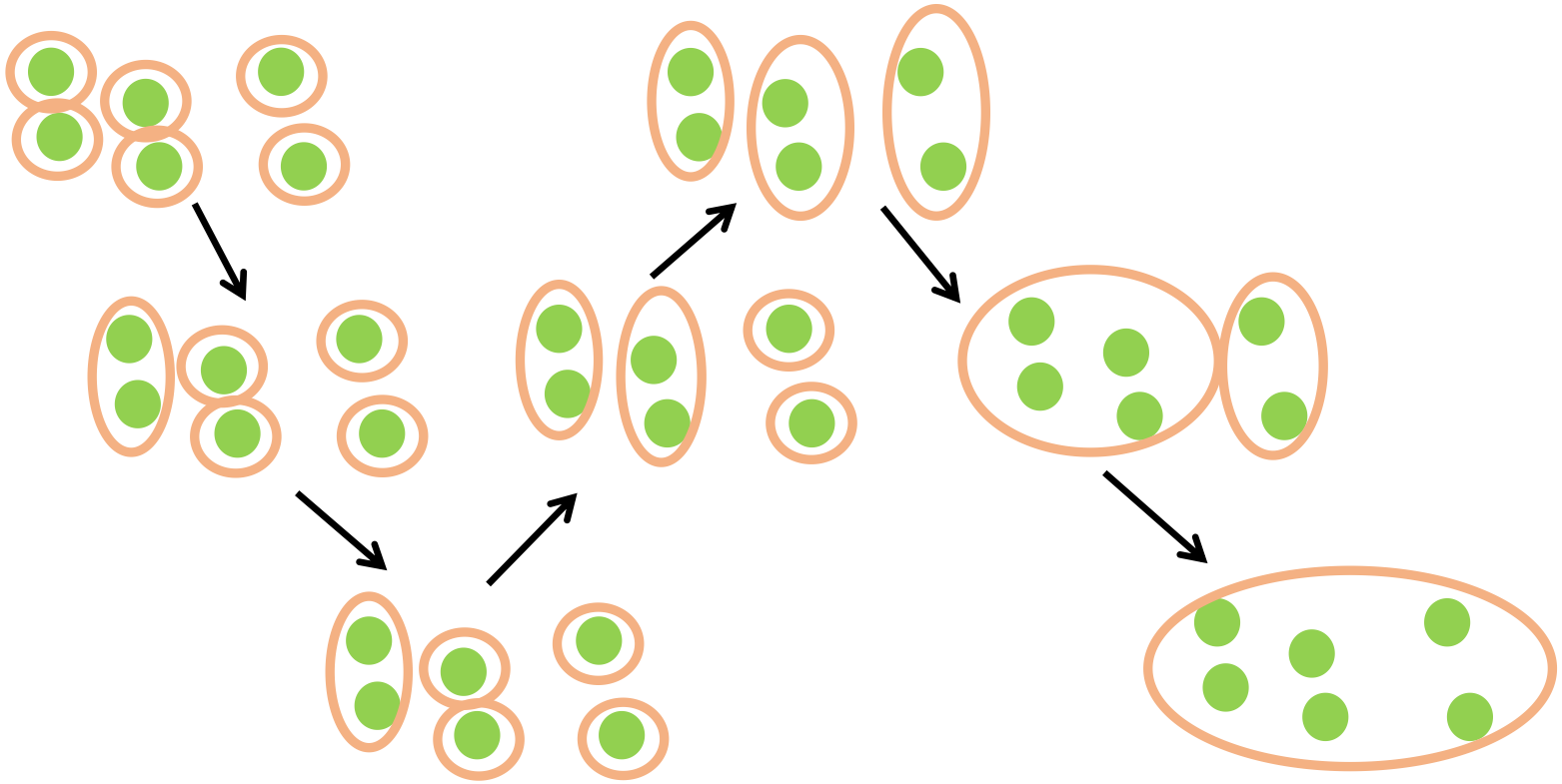


Hierarchical Clustering

Idea

Goal: Train a **hierarchy** of clusters

- In **Agglomorative Clustering** we start with each node having its own cluster and then iteratively combine clusters.

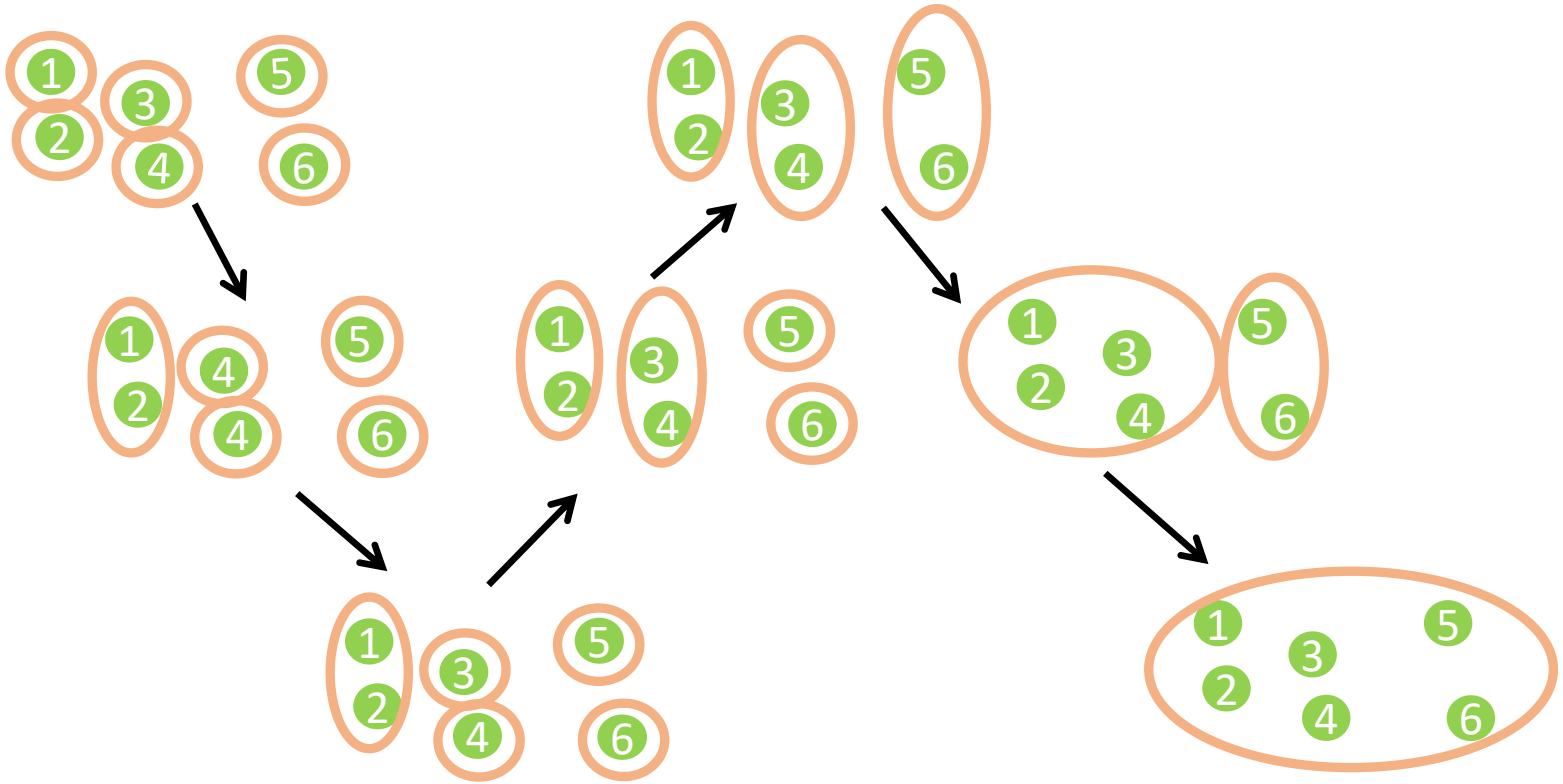


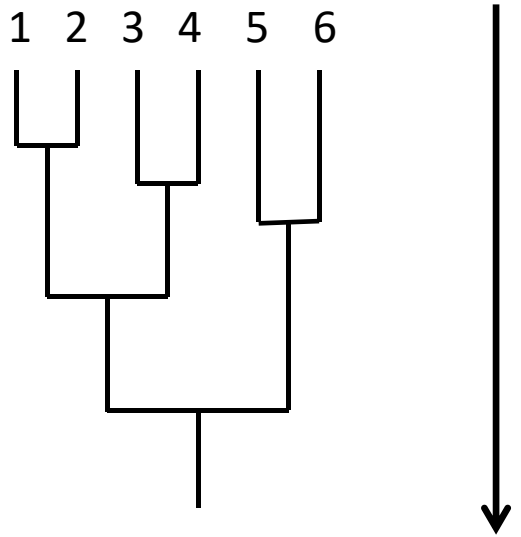
Hierarchical Clustering

Idea

Goal: Train a **hierarchy** of clusters

- In **Agglomorative Clustering** we start with each node having its own cluster and then iteratively combine clusters.



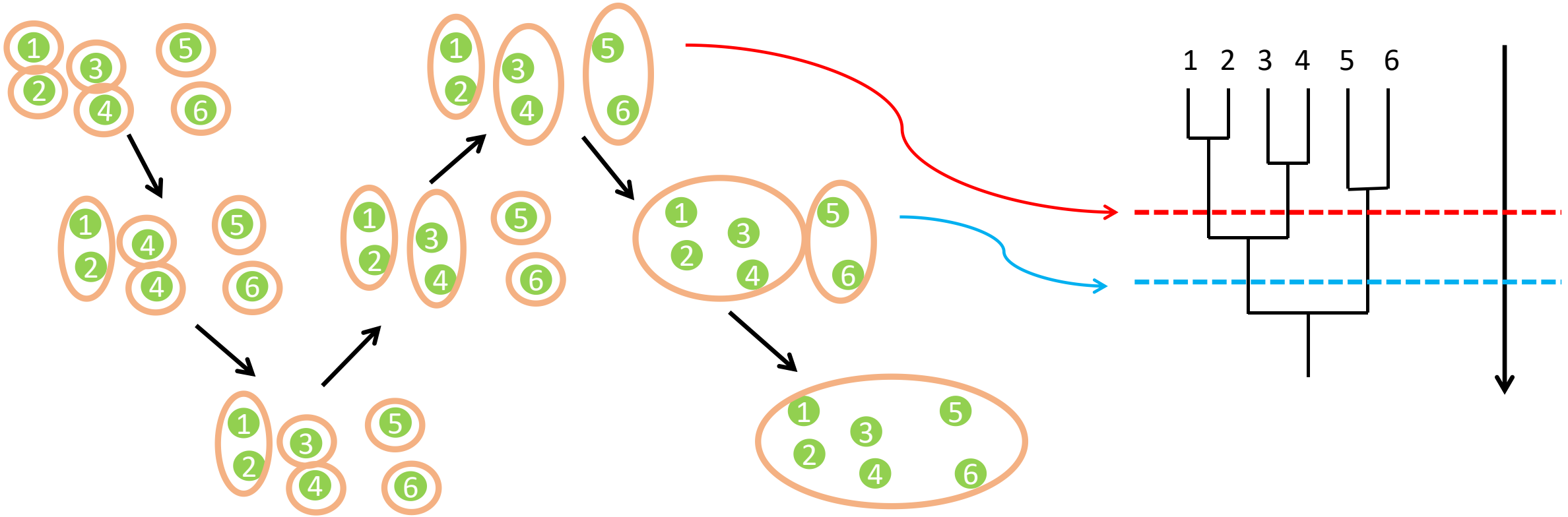


Hierarchical Clustering

Idea

Goal: Train a **hierarchy** of clusters

- In **Agglomorative Clustering** we start with each node having its own cluster and then iteratively combine clusters.

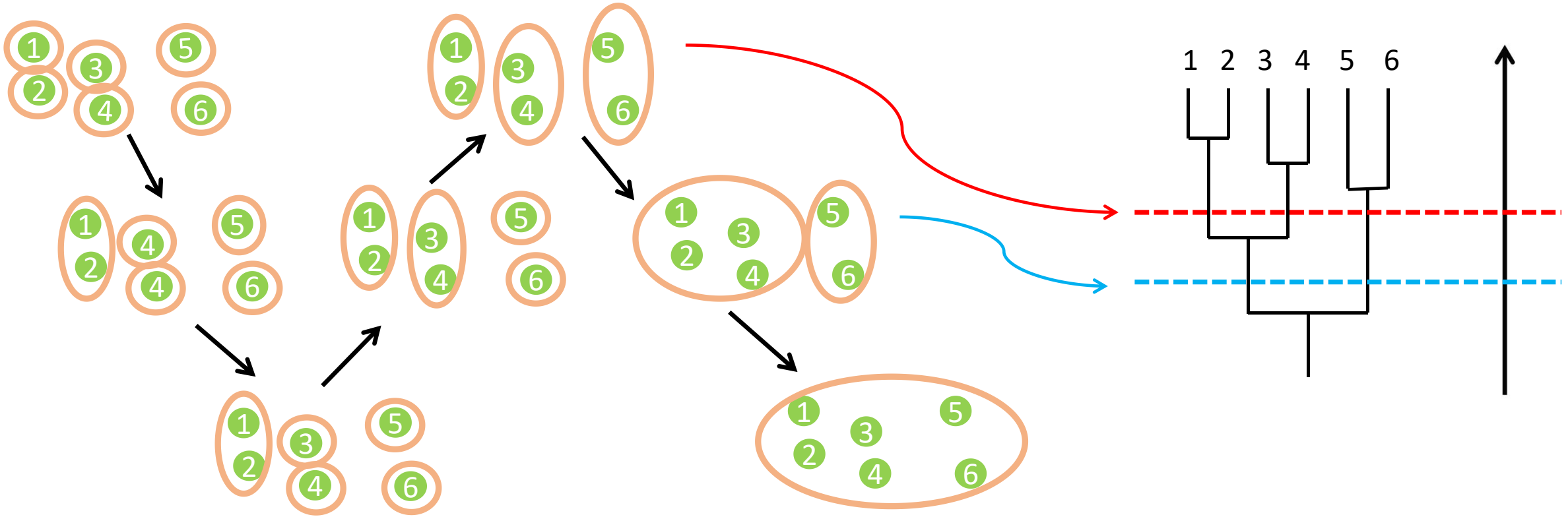


Hierarchical Clustering

Idea

Goal: Train a **hierarchy** of clusters

- In **Divisive Clustering** we start with all nodes in one cluster and then iteratively divide clusters.



Agglomerative Clustering

How it works

Agglomerative Clustering Algorithm:

- Define a distance measure between data points.
- Define a distance measure between clusters.
- Initialize: Put every example into its own cluster.
- Until all nodes in one cluster do
    - Select the two clusters with minimal distance
    - Join these two clusters

Agglomerative Clustering

How it works

Agglomerative Clustering Algorithm:

- Define a distance measure between data points.
- Define a distance measure between clusters.
- Initialize: Put every example into its own cluster.
- Until all nodes in one cluster do
  - Select the two clusters with minimal distance
  - Join these two clusters

- Minimal distance

$$D(A,B) := \min_{a \in A, b \in B} \{d(a,b)\}$$

- Maximal distance

$$D(A,B) := \max_{a \in A, b \in B} \{d(a,b)\}$$

- Mean distance

$$D(A,B) := \frac{1}{|A||B|} \sum_{a \in A, b \in B} \{d(a,b)\}$$

HDBSCAN

How it works

4 steps of HDBSCAN

1. Define new distance function according to the density/sparsity.
2. Perform hierarchical clustering with metric from first step.
3. Condense cluster hierarchy based on minimum cluster size.
4. Extract stable clusters

HDBSCAN

How it works

4 steps of HDBSCAN

1. Define new distance function according to the density/sparsity.
2. Perform hierarchical clustering with metric from first step.
3. Condense cluster hierarchy based on minimum cluster size.
4. Extract stable clusters

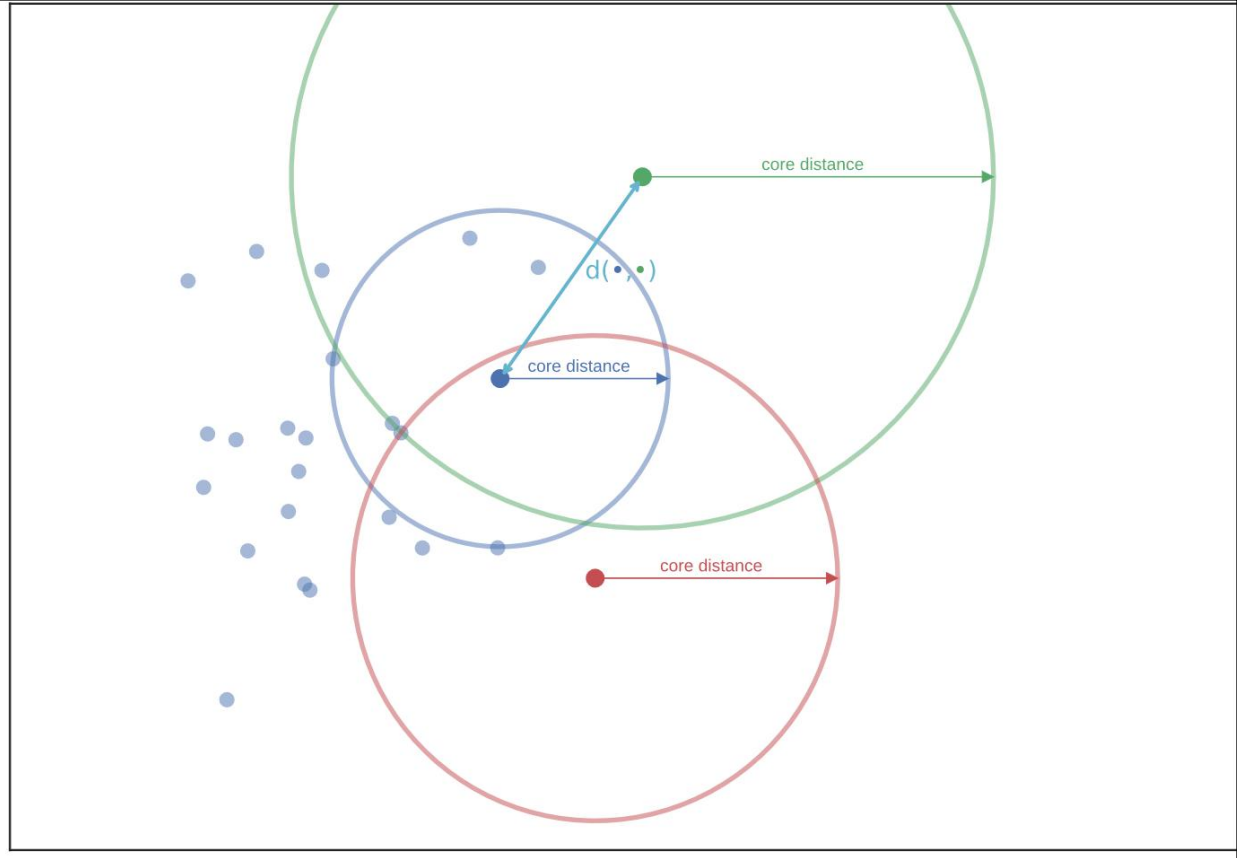
From old metric d, define **mutual reachibility distance** by

$$d(a,b)=\max\{ r_k(a), r_k(b), d(a,b)\}$$

where $r_k(a)$=distance to k-th closest neighbour
= „core distance of a“

HDBSCAN

How it works



4 steps of HDBSCAN

1. Define new distance function according to th
2. Perform hierarchical clustering with metric fr
3. Condense cluster hierarchy based on minimum cluster size
4. Extract stable clusters

From old metric d, define **mutual reachability distance** by

$$d(a,b)=\max\{ r_k(a), r_k(b), d(a,b)\}$$

where $r_k(a)$=distance to k-th closest neighbour
= „core distance of a“

HDBSCAN

How it works

4 steps of HDBSCAN

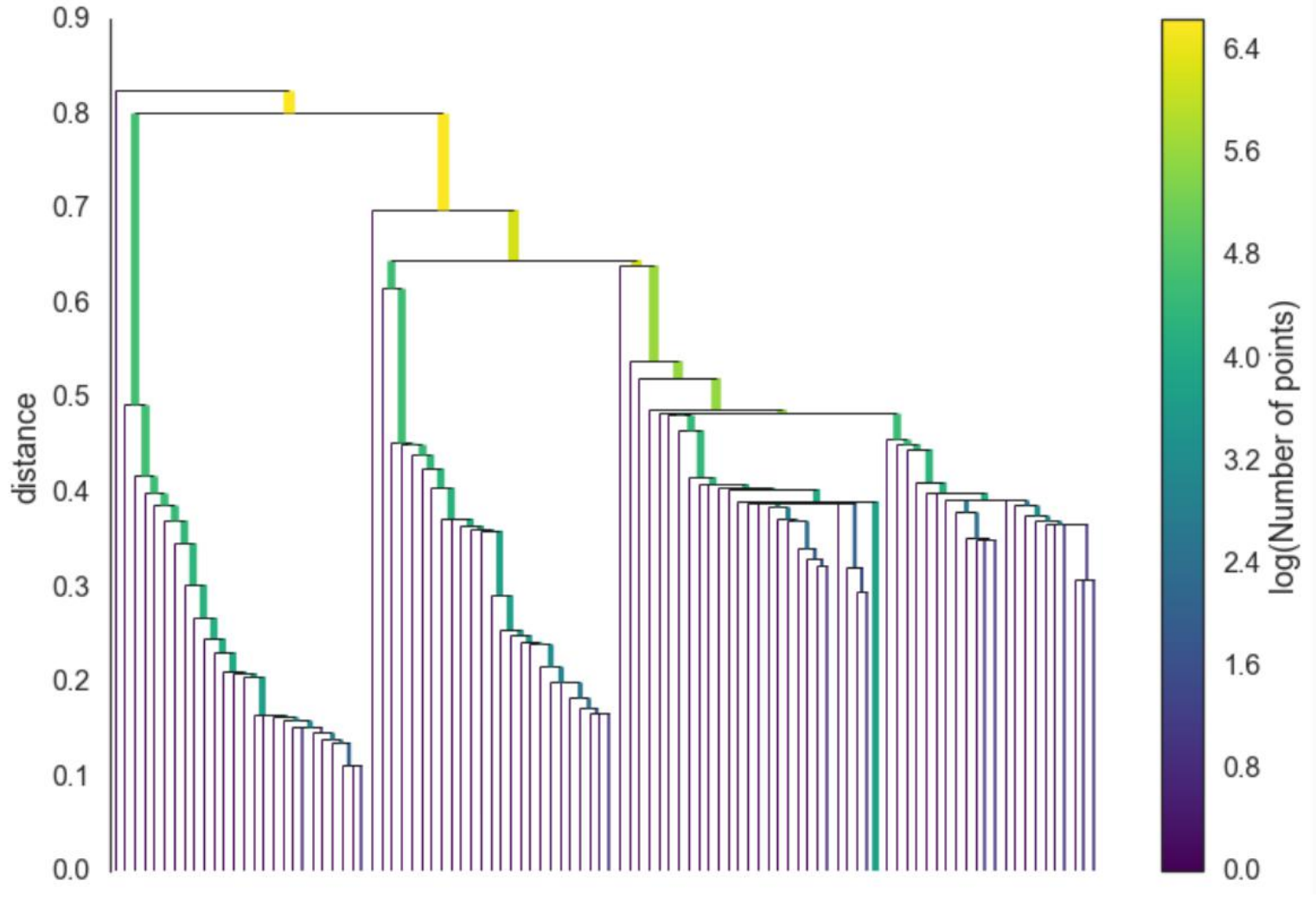
1. Define new distance function according to the density/sparsity.
2. Perform hierarchical clustering with metric from first step.
3. Condense cluster hierarchy based on minimum cluster size.
4. Extract stable clusters

HDBSCAN

How it works

4 steps of HDBSCAN

1. Define new distance function a
2. Perform hierarchical clustering
3. Condense cluster hierarchy bas
4. Extract stable clusters



HDBSCAN

How it works

4 steps of HDBSCAN

1. Define new distance function according to the density/sparsity.
2. Perform hierarchical clustering with metric from first step.
3. Condense cluster hierarchy based on minimum cluster size.
4. Extract stable clusters

Only consider the split of clusters where both clusters after splitting have more nodes than specified by the parameter „min number of cluster size“. Otherwise, throw nodes out.

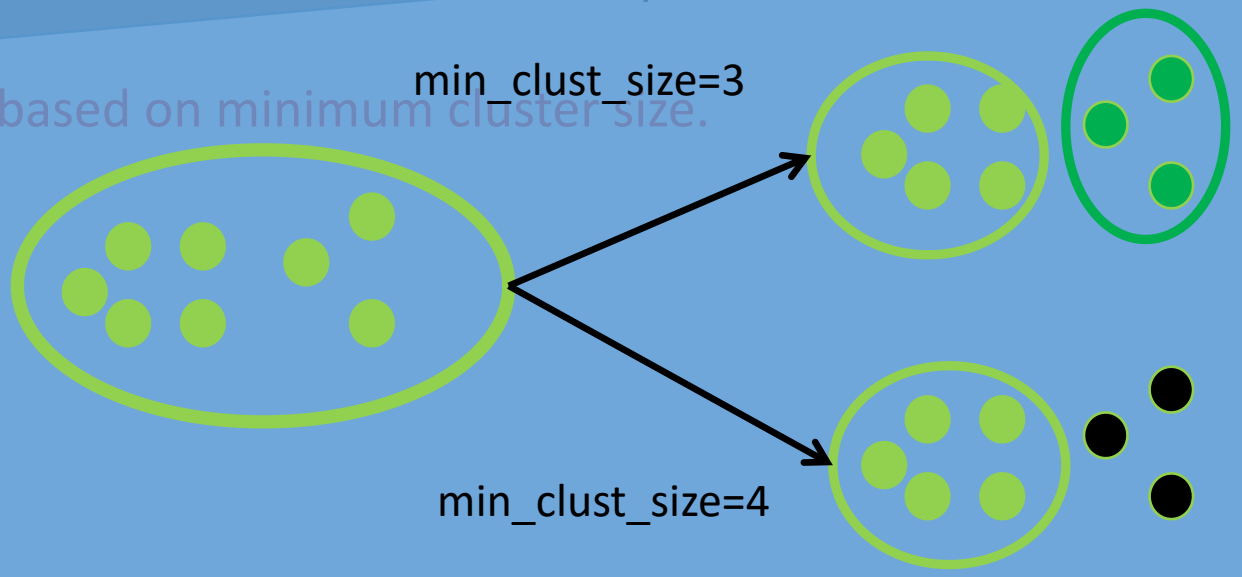
HDBSCAN

How it works

4 steps of HDBSCAN

1. Define new distance function according to the density/sparsity.
2. Perform hierarchical clustering on the transformed space.
3. Condense cluster hierarchy based on minimum cluster size.
4. Extract stable clusters

Only consider the split of clusters where both clusters after splitting have more nodes than specified by the parameter „min number of cluster size“. Otherwise, throw nodes out.



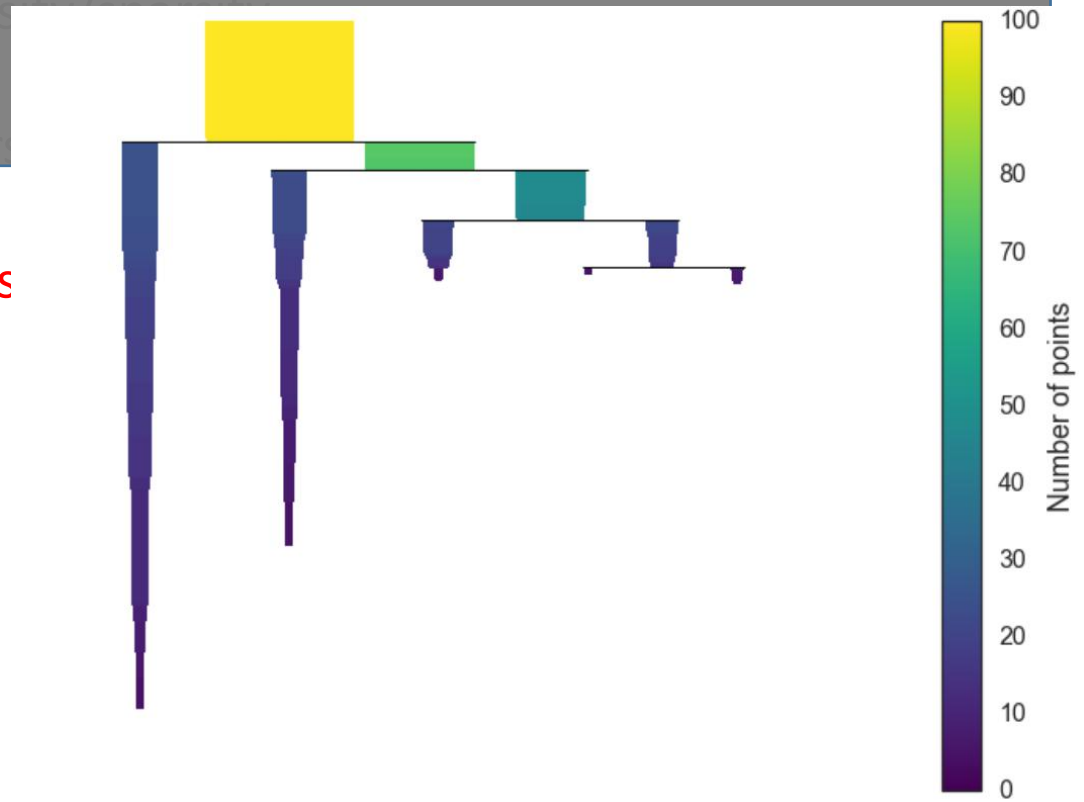
HDBSCAN

How it works

Only consider the split of clusters where both clusters after splitting have more nodes than specified by the parameter „min number of cluster size“. Otherwise, throw nodes out.

4 steps of HDBSCAN

1. Define new distance function according to the density
2. Perform hierarchical clustering with metric from first step
3. Condense cluster hierarchy based on minimum cluster size
4. Extract stable clusters



HDBSCAN

How it works

4 steps of HDBSCAN

1. Define new distance function according to the density/sparsity.
2. Perform hierarchical clustering with metric from first step.
3. Condense cluster hierarchy based on minimum cluster size.
4. Extract stable clusters

Consider a measure of persistency of clusters defined by the distance values at which the cluster starts, at which it ends and at when points fall out of the cluster.
Take the cluster with the largest stability values.

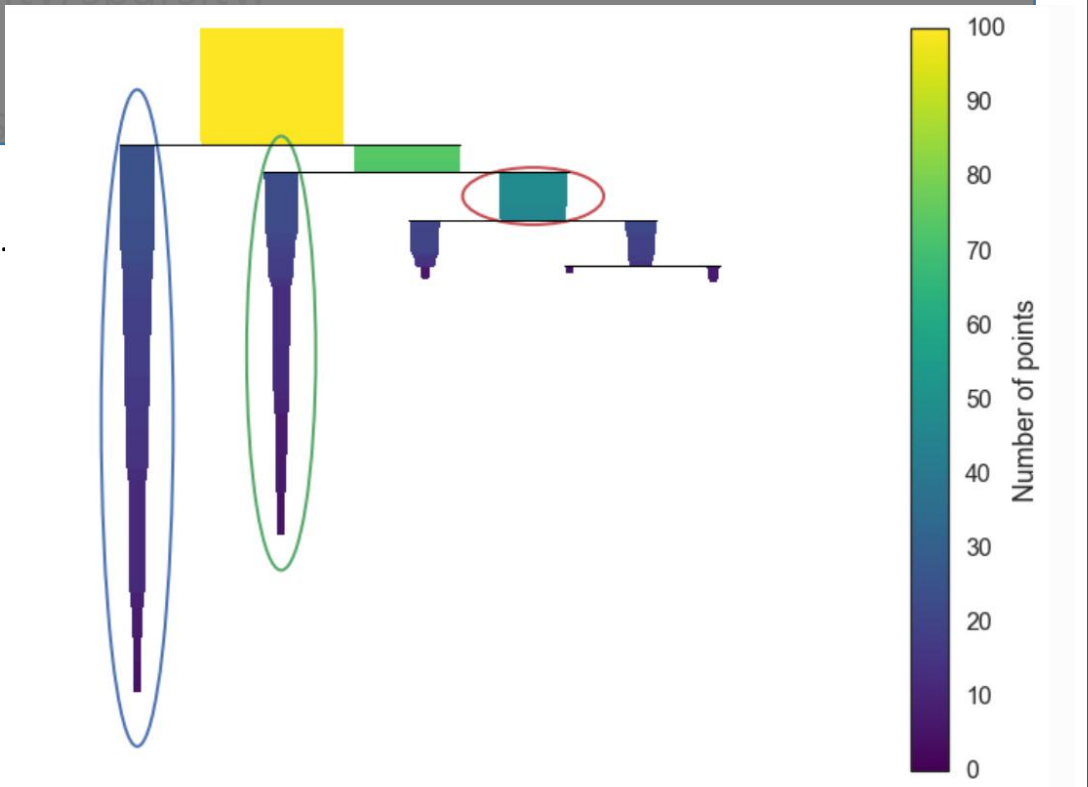
HDBSCAN

How it works

Consider a measure of persistency of clusters defined by the distance values at which the cluster starts, at which it ends and at when points fall out of the cluster.
Take the cluster with the largest stability values.

4 steps of HDBSCAN

1. Define new distance function according to the density/sparsity.
2. Perform hierarchical clustering with metric from first
3. Condense cluster hierarchy based on minimum clus
4. Extract stable clusters



Clustering methods

Overview

Which approach should I use?

- k-means can be used as default

- In the case of
  - existence of outliers
  - complex shapes
  - unclear cluster sizes

  then use DBSCAN, and if
  - clusters have very different densities

  then use HDBSCAN

- There are many more clustering techniques!

Clustering methods

Overview

Which approach should I use?

- k-means can be used as default
- In the case of
  - existence of outliers
  - complex shapes
  - unclear cluster sizes

  then use DBSCAN, and if
  - clusters have very different densities

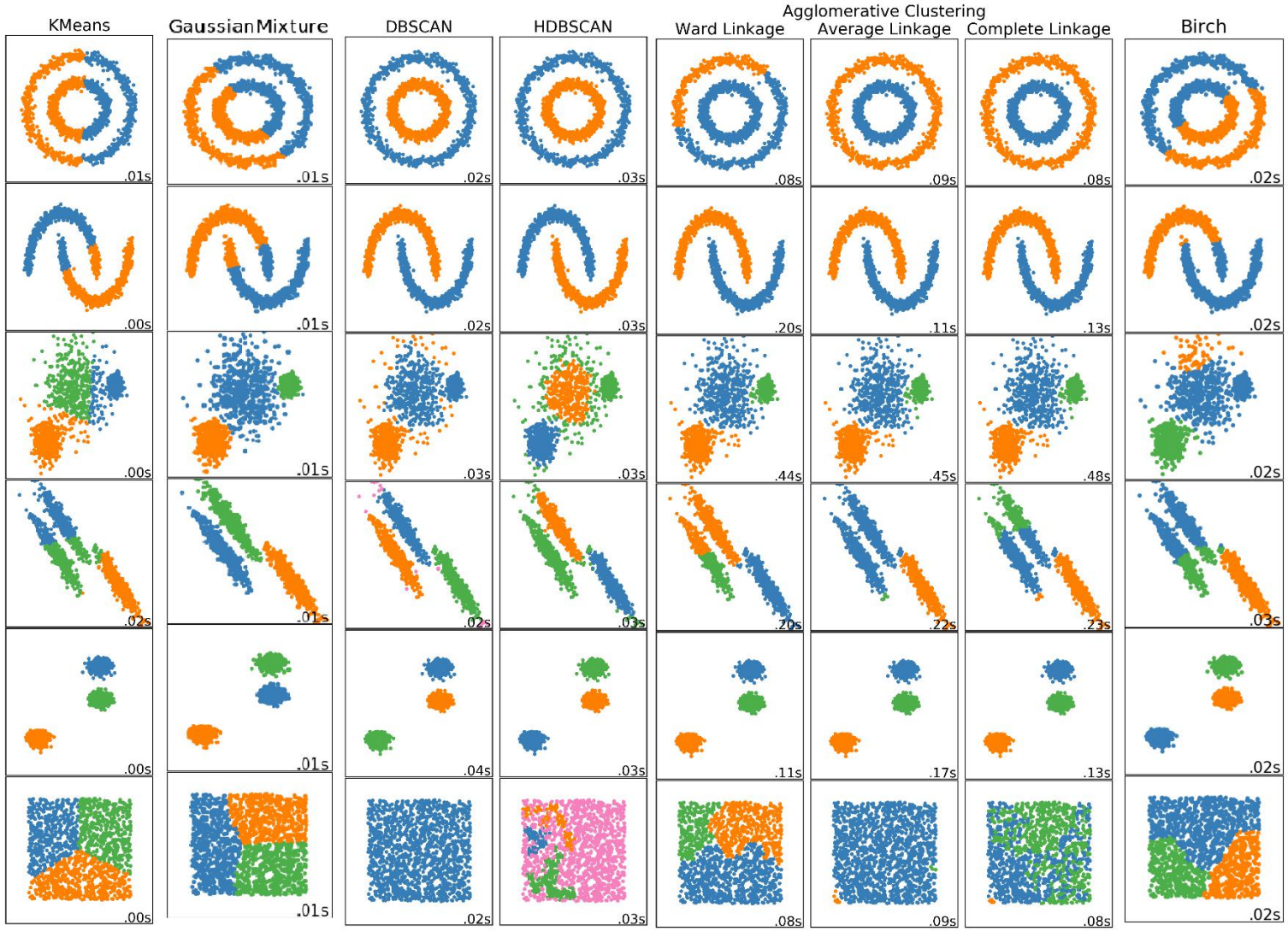
  then use HDBSCAN
- There are many more clustering techniques!

What if the dataset is huge?

- K-means and HDBSCAN are very fast, but still too expensive for the big data regime.
- Note that each iteration of k-means requires a sweep over all data points
- Use k-means with mini batches (each iteration step only uses a small subset of data) or use a clustering algorithm called Birch (not discussed here)

Clustering methods

Overview



Clustering methods

Distance is key

- Note that all clustering algorithm depend on the metrix d(x,y) between two points x and y.
- In our examples in 2 dimensions we have always considered d to be the Euclidean metric.
- Sometimes other metrics make more sense.
- Changing the metric d implies different results for the clustering algorithm!! So a good manual choice for d is necessary.

Evaluation

Comparing different cluster algorithms

- **Evaluation** of clustering algorithms is hard, since also what means to be a „good cluster“ depends on interpretation and the choice of metric.

- Having fixed a metric and algorithm, we sometimes still want to evaluate the different outcomes for different parameters, for example
  - the number of clusters K in -means
  - the neighborhood parameter ε in DBSCAN

- We will discuss two such methods now

Evaluation

Silhouette Values

The Silhouette value computes for each point a value in [-1,1] indicating how well the point fits into its cluster

$$\mathcal{C}_i = \{x_k \text{ in the same cluster as } x_i\}$$

$$\mathcal{A}_i = \{x_k \text{ in the next closest cluster to the cluster of } x_i\}$$

$$c(x_i) = \frac{1}{|\mathcal{C}_i|} \sum_{k \in \mathcal{C}_i} d(x_k, x_i)$$

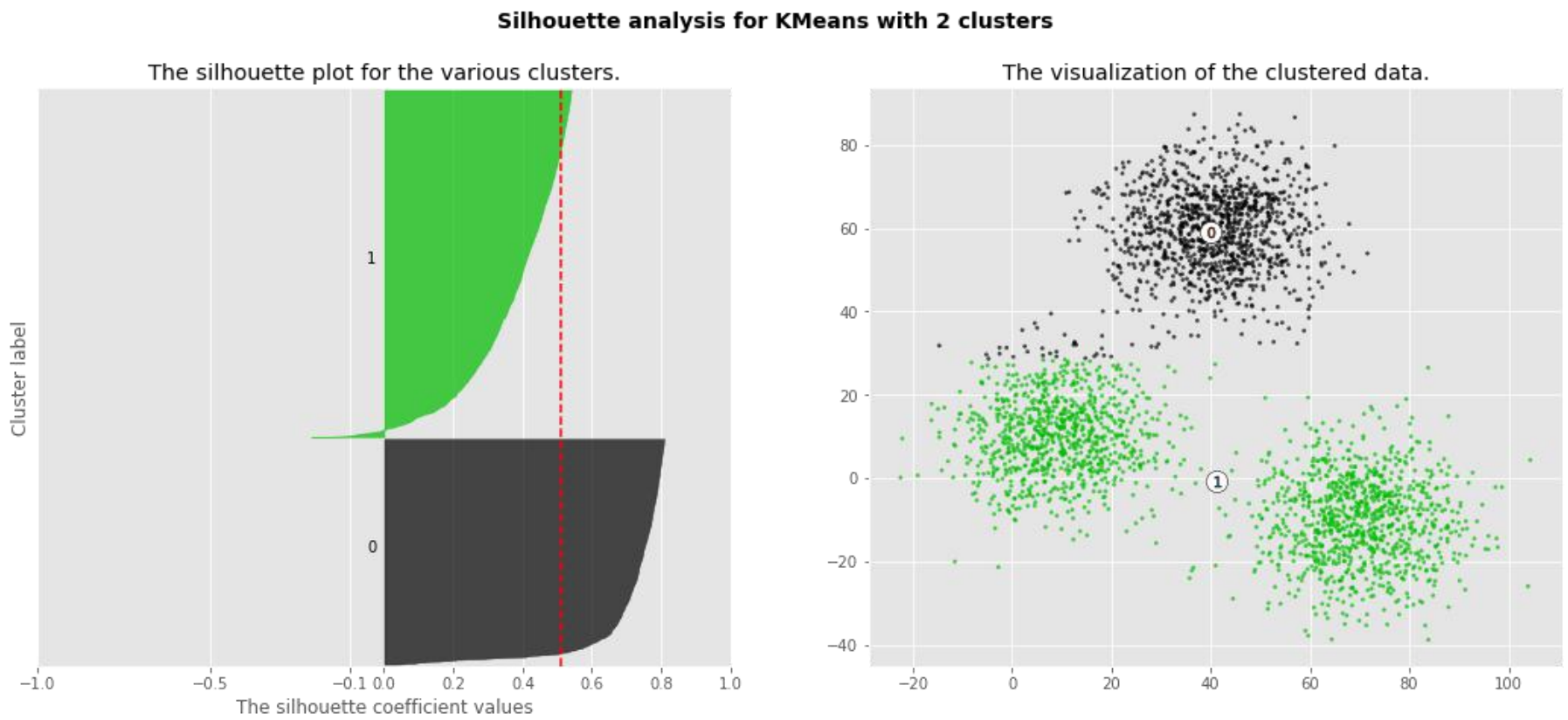
$$a(x_i) = \frac{1}{|\mathcal{A}_i|} \sum_{k \in \mathcal{A}_i} d(x_k, x_i)$$

$$S(x_i) = \frac{a(x_i) - c(x_i)}{\max\{c(x_i), a(x_i)\}}$$

Evaluation

Silhouette Values

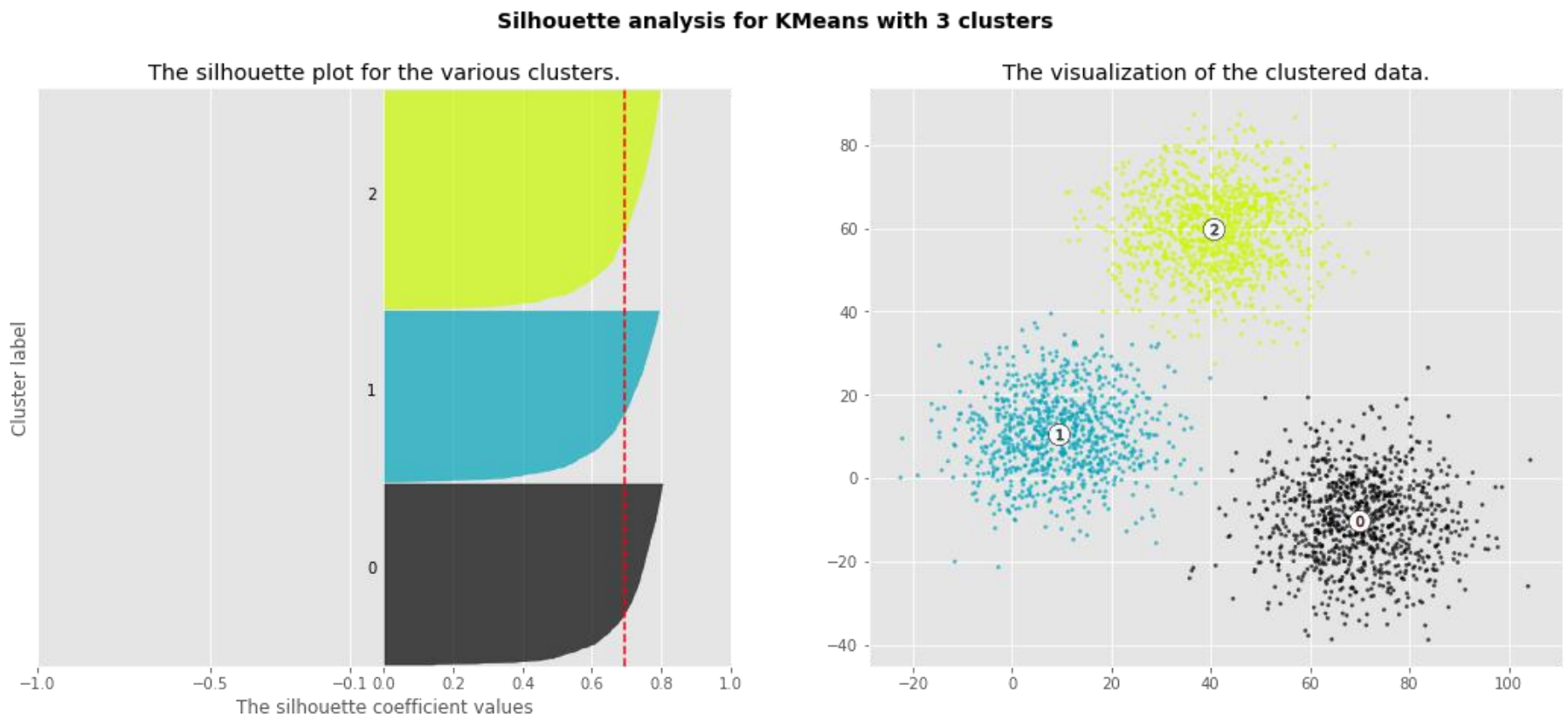
- Silhouette plots visualize the Silhouette value for each point.
- By computing the average of the Silhouette values over all points we reduce to a single value to compare different clustering methods (indicated by red line).



Evaluation

Silhouette Values

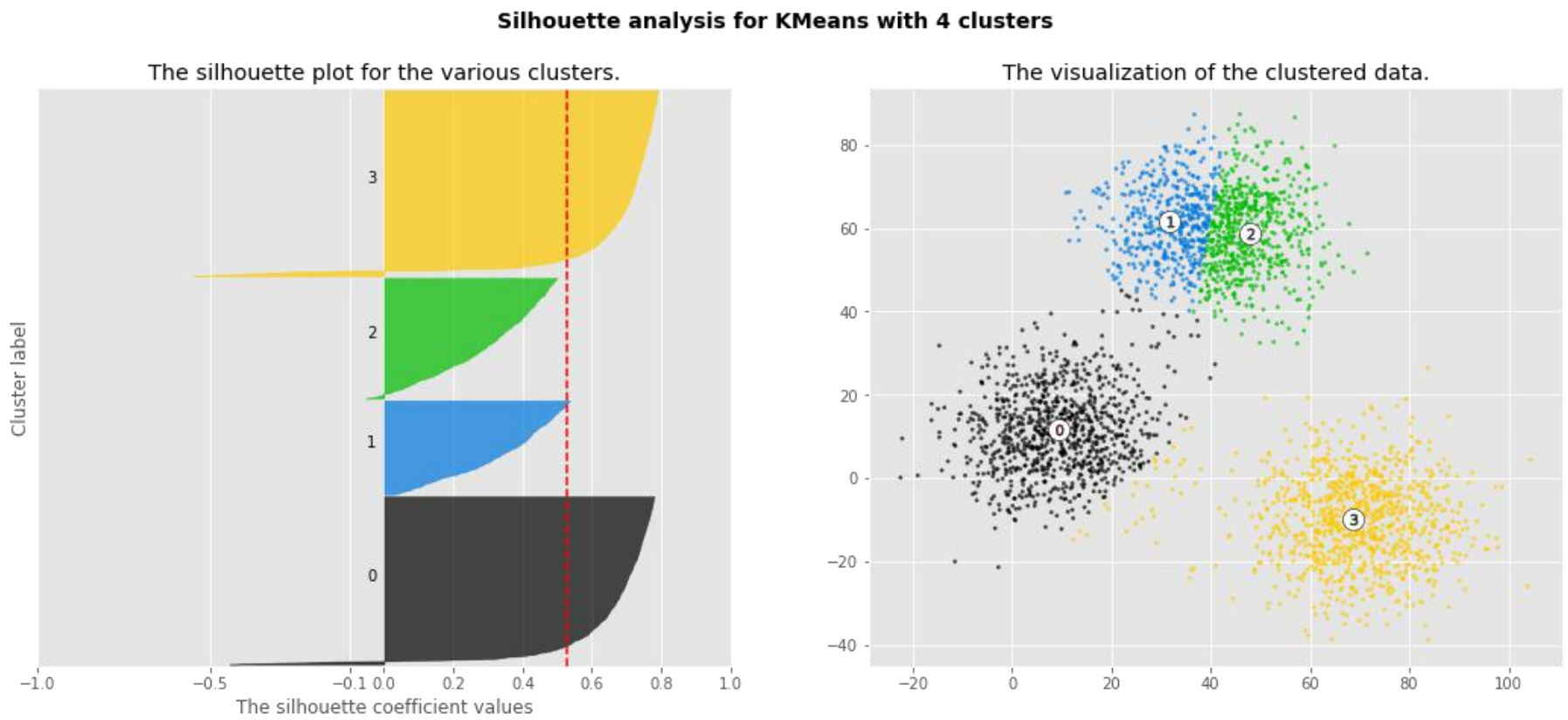
- Silhouette plots visualize the Silhouette value for each point.
- By computing the average of the Silhouette values over all points we reduce to a single value to compare different clustering methods (indicated by red line).



Evaluation

Silhouette Values

- Silhouette plots visualize the Silhouette value for each point.
- By computing the average of the Silhouette values over all points we reduce to a single value to compare different clustering methods (indicated by red line).



Evaluation

By objective

Clustering methods have an **associated objective funtion**, which we can use for evaluation to find good hyperparameter choices.

Example: K-Means

- **Objective function:** $$\min_{c_1,\ldots,c_n,\mu_1,\ldots,\mu_k} \sum_{i=1}^{n} ||x_i - \mu_{c_i}||^2$$

$$\mu_i = \text{ centroid of i-th cluster}$$
$$c_i = \text{ cluster assignment to point i}$$

- The two **alternating steps of the K-Means** algorithm correspond to iterative minimization over c_i and μ_j

Evaluation
By objective

Clustering methods have an **associated objective funtion**, which we can use for evaluation to find good hyperparameter choices.

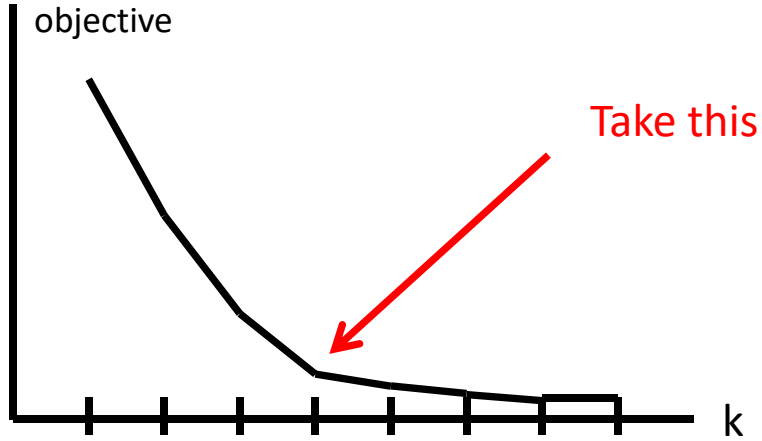
Example: K-Means

- **Objective function:** $\min_{c_1,\ldots,c_n,\mu_1,\ldots,\mu_k} \sum_{i=1}^{n} ||x_i - \mu_{c_i}||^2$

$\mu_i =$ centroid of i-th cluster

$c_i =$ cluster assignment to point i

- The two **alternating steps of the k-Means** algorithm correspond to iterative minimization over c_i and μ_j
- Use the **elbow point** to find the best k



Evaluation
By application

Note that both **evaluation methods involve** a chosen **metric**.

What is a **good cluster** again **depends on a chosen distance**.

Which distance is good depends on **application**.

Market Segmentation

Group your customers into groups of interest

In general, an **evaluation** of good clusters or an appropriate number of clusters is **given by the application**

Example:
If I produce clothes and I want to cluster my customers into different sizes, then I might want to cluster into 5 clusters: XS, S, M, L, XL

Dimensionality Reduction

Representation Learning

Dimensionality reduction

Unsupervized learning targets to learn a **new representation** of data that contains **information** on **structure of the given data**.

Dimensionality Reduction when new representation in smaller dimension, e.g. when data lies in $\mathbb{R}^n$ and we find a new representation

$$\pi : \mathbb{R}^n \to \mathbb{R}^m, \ m < n$$

Representation Learning

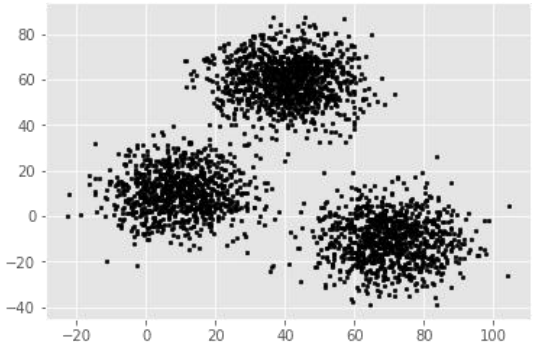
Dimensionality reduction

Unsupervized learning targets to learn a **new representation** of data that contains **information** on **structure of the given data**.

Dimensionality Reduction when new representation in smaller dimension, e.g. when data lies in $\mathbb{R}^n$ and we find a new representation

$$\pi : \mathbb{R}^n \to \mathbb{R}^m, \ m < n$$

Clustering is an extreme case where m=0, every point is represented by its cluster or cluster centroid c_i



$$\longrightarrow \quad \{1, 2, \ldots, k\} \cong \{c_1, c_2, \ldots, c_k\}$$

Representation Learning

K-Means as data representation

- K means

Points $x_i \in \mathbb{R}^n$

$D_k = \mathbb{R}^{n \times k}$ matrices

$\phi : \mathbb{R}^n \to \{1, 2, \ldots, k\}$

$V_j = \{x \in \mathbb{R}^n \mid \phi(x) = j\}$

Objective:

$$\min_{\mu \in D_k} \sum_{j=1}^{k} \sum_{x_i \in V_j} ||x_i - \mu_j||^2 \text{ where } \mu_j \text{ is the } j\text{-th column of } \mu$$

Note that minimizing for μ results in the mean of all x_i in V_j

Dimensionality Reduction

PCA

- **PCA= Principal Compenent Analysis** is the most popular dimension reduction technique

$$\pi : \mathbb{R}^n \to \mathbb{R}^m, \ m < n$$

- PCA finds the best subspace of a given dimension m in the sense of least-square error.

$$x \approx \sum_j \alpha_j w_j \text{ for orthonormal vectors } w_j$$

More precisely, we are searching for m vectors w_j and coefficients $\alpha_j(x)$ dependent on x such that

$$||x_i - \sum_j \alpha_j(x_i) w_j||^2 \text{ minimal in expectation over all samples } x_i$$

Dimensionality Reduction

PCA

- For PCA, assume that the mean of samples x_i is zero

You need to normalize your data to zero mean when you want to apply PCA!

- Since the vectors w_i are assumed to be orthonormal (i.e. mutually orthogonal of of unit length), the best coefficents are given by

$$\alpha_j(x) = \langle x, w_j \rangle$$

- The **objective of PCA** is therefore to find m vectors w_j in $\mathbb{R}^n$ (m <n) such that

$$\frac{1}{N}\sum_{i=1}^{N} ||x_i - \sum_{j=1}^{m} \langle x_i, w_j \rangle w_j||^2 \text{ is minimal}$$

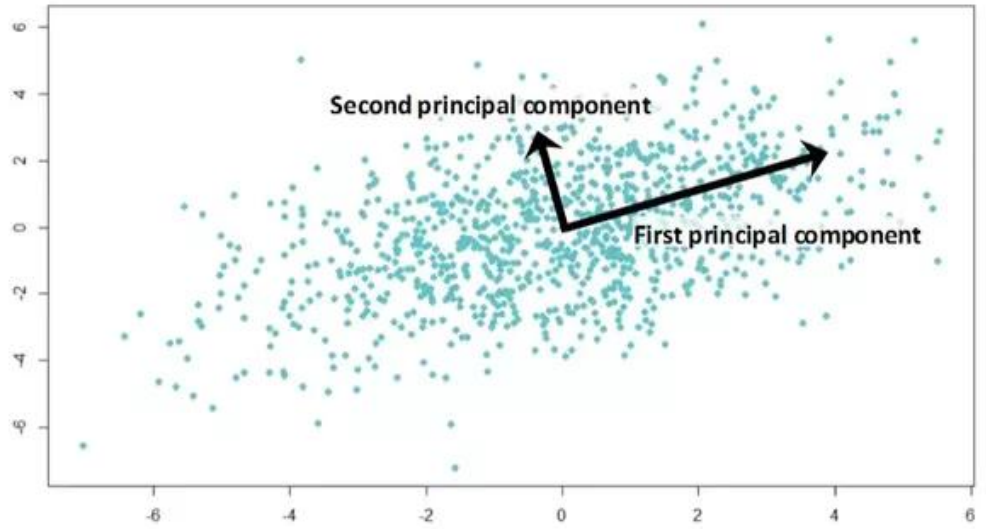
Dimensionality Reduction

PCA

- The name „Principal Component Analysis“ comes from the fact that the solution to the objective of PCA equals the subspace spanned by the first m prinicipal components of the data, i.e, the m orthogonal directions of largest variance.

- In particular, w_1 = first principal component = the direction of largest variance in the data

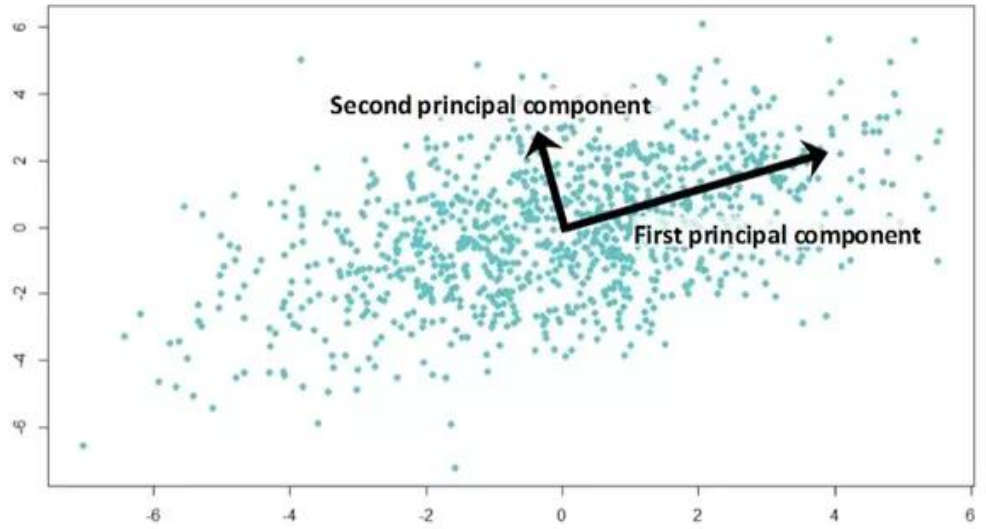
- We will make this connection more precise later.



Dimensionality Reduction

PCA

- The name „Principal Component Analysis“ comes from the fact that the solution to the objective of PCA equals the subspace spanned by the first m prinicipal components of the data, i.e, the m orthogonal directions of largest variance.
- In particular, w_1 = first principal component = the direction of largest variance in the data
- We will make this connection more precise later.
- First, recap of variance and covariance



Statistic Recap

Sample Variance and Covariance

$$\mathcal{X} = \{x_1, x_2, \ldots, x_N\} \text{ set of samples in } \mathbb{R}^n$$

The sample mean of $\mathcal{X}$ is given by $\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i$. Recall that we assume $\bar{x} = 0$.

If a pair of random variables (S, T) can take on values (s_i, t_i) with equal probabiliy, then the covariance is defined by $\frac{1}{N}\sum_i (s_i - \bar{s})(t_i - \bar{t})$

We can compute the covariance between all dimension of our vector-valued samples and store the results in the so-called covariance matrix.

With $\bar{x} = 0$, this gives the matrix $C = \frac{1}{N} X X^T$ where $X = [x_1 | x_2 | \ldots | x_N] \in \mathbb{R}^{n \times N}$.

$$\text{Covariance matrix } C = \frac{1}{N}\, X \cdot X^T = \frac{1}{N} \sum_{i=1}^{N} x_i \cdot x_i^T \in \mathbb{R}^{n \times n}$$

PCA

Covariance matrix and the spectral theorem

$$\text{Covariance matrix } C = \frac{1}{N}\, X \cdot X^T = \frac{1}{N} \sum_{i=1}^{N} x_i \cdot x_i^T \in \mathbb{R}^{n \times n}$$

The **covariance matrix** is a symmetric positive definite matrix. By the **spectral theorem**, C has an orthonormal basis

$$\{w_1, w_2, \ldots, w_n\} \text{ of eigenvectors with eigenvalues } \lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n \geq 0$$

Note that we have ordered the eigenvalues by size.

Every sample can therefore be written as

$$x_i = \sum_{j=1}^{n} \langle x_i, w_j \rangle w_j .$$

PCA

Covariance matrix and the spectral theorem

$$\text{Covariance matrix } C = \frac{1}{N}\, X \cdot X^T = \frac{1}{N} \sum_{i=1}^{N} x_i \cdot x_i^T \in \mathbb{R}^{n \times n}$$

The **covariance matrix** is a symmetric positive definite matrix. By the **spectral theorem**, C has an orthonormal basis

$$\{w_1, w_2, \ldots, w_n\} \text{ of eigenvectors with eigenvalues } \lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n \geq 0$$

Note that we have ordered the eigenvalues by size.

Every sample can therefore be written as

$$x_i = \sum_{j=1}^{n} \langle x_i, w_j \rangle w_j .$$

$$\frac{1}{N} \sum_{i=1}^{N} ||x_i - \sum_{j=1}^{m} \langle x_i, w_j \rangle w_j||^2 \text{ is minimal}$$

This looks already close to the form in our objective of PCA. We only need to make sure the eigenvectors are ideal for our objective and to take out only m<n of them.

PCA

Covariance matrix and the spectral theorem

$$\text{Covariance matrix } C = \frac{1}{N}\, X \cdot X^T = \frac{1}{N} \sum_{i=1}^{N} x_i \cdot x_i^T \in \mathbb{R}^{n \times n}$$

The **covariance matrix** is a symmetric positive definite matrix. By the **spectral theorem**, C has an orthonormal basis

$$\{w_1, w_2, \ldots, w_n\} \text{ of eigenvectors with eigenvalues } \lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n \geq 0$$

Theorem: Let w_1 be the eigenvector belonging to the largest eigenvalue of C. Then the variance of the data is the largest into the direction of w_1 .

PCA

Covariance matrix and the spectral theorem

$$\text{Covariance matrix } C = \frac{1}{N}\, X \cdot X^T = \frac{1}{N} \sum_{i=1}^{N} x_i \cdot x_i^T \in \mathbb{R}^{n \times n}$$

Theorem: Let w_1 be the eigenvector belonging to the largest eigenvalue of C. Then the variance of the data is the largest into the direction of w_1 .

Theorem: Let w_1 be the eigenvector of C with largest eigenvalue defining the direction of largest variance of the data.

Then

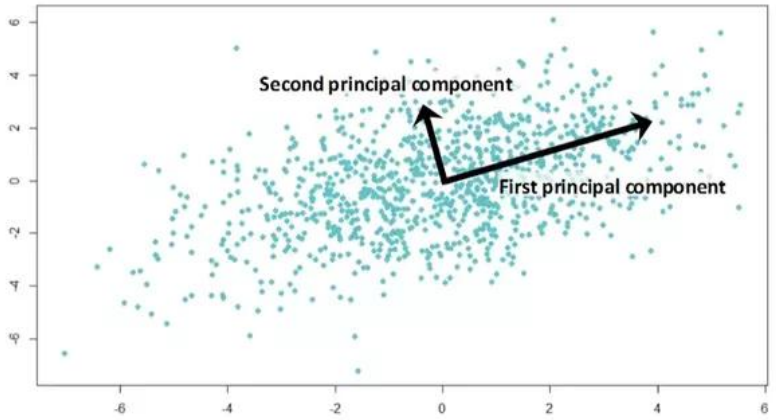
$$\frac{1}{N} \sum_{i=1}^{N} ||x_i - \langle x_i, w \rangle w||^2 \text{ is minimal for } w = w_1$$

PCA

Covariance matrix and the spectral theorem

$$\text{Covariance matrix } C = \frac{1}{N}\, X \cdot X^T = \frac{1}{N} \sum_{i=1}^{N} x_i \cdot x_i^T \in \mathbb{R}^{n \times n}$$

Theorem: Let w_1 be the eigenvector belonging to the largest eigenvalue of C. Then the variance of the data is the largest into the direction of w_1 .



PCA

Covariance matrix and the spectral theorem

$$\text{Covariance matrix } C = \frac{1}{N} X \cdot X^T = \frac{1}{N} \sum_{i=1}^{N} x_i \cdot x_i^T \in \mathbb{R}^{n \times n}$$

The first principal component w_1

= direction w_1 of largest variance in data

= normalized eigenvector w_1 of covariance matrix with largest eigenvalue

= normalized direction w_1 such that $||x - \langle x, w_1 \rangle w_1||^2$ is minimal

Theorem: Let w_1 be the eigenvector belonging to the largest eigenvalue of C. Then the variance of the data is the largest into the direction of w_1 .

Theorem: Let w_1 be the eigenvector defining the direction of largest variance of the data. Then

PCA

Covariance matrix and the spectral theorem

$$\text{Covariance matrix } C = \frac{1}{N} X \cdot X^T = \frac{1}{N} \sum_{i=1}^{N} x_i \cdot x_i^T \in \mathbb{R}^{n \times n}$$

The first *m* principal components $w_1, w_2, .., w_m$

= the *m* directions $w_1, w_2, .., w_m$ of largest variance in data

= *m* normalized eigenvectors $w_1, w_2, .., w_m$ of covariance matrix with largest eigenvalues

= *m* normalized directions $w_1, w_2, .., w_m$ such that $||x - \Sigma_j \langle x, w_j \rangle w_j||^2$ is minimal

Theorem: Let w_1 be the eigenvector belonging to the largest eigenvalue of C.
Then the variance of the data is the largest into the direction of w_1 .

Theorem: Let w_1 be the eigenvector defining the direction of largest variance of the data.
Then

PCA

Representation in PCA space

Conclusion:

We compute the eigenvectors and eigenvalues of the

$$\text{Covariance matrix } C = \frac{1}{N}\, X \cdot X^T = \frac{1}{N} \sum_{i=1}^{N} x_i \cdot x_i^T \in \mathbb{R}^{n \times n}$$

Choose the first *m* eigenvectors corresponding to the *m* largest eigenvalues.

Then for all samples x from the distribution, we have

$$x \approx \sum_{j=1}^{m} \langle x, w_j \rangle w_j$$

PCA

Representation in PCA space

Conclusion:

We compute the eigenvectors and eigenvalues of the

$$\text{Covariance matrix } C = \frac{1}{N}\, X \cdot X^T = \frac{1}{N} \sum_{i=1}^{N} x_i \cdot x_i^T \in \mathbb{R}^{n \times n}$$

Choose the first *m* eigenvectors corresponding to the *m* largest eigenvalues.

Then for all samples x from the distribution, we have

$$x \approx \sum_{j=1}^{m} \langle x, w_j \rangle w_j$$

Within the m-dimensional subspace $\quad \text{span}\{w_j \mid j = 1, 2, \ldots, m\}$

x is encoded by the coefficients $\langle x, w_j \rangle$

$\Rightarrow$ The dimensionality reduction is defined by the representation $\pi : \mathbb{R}^n \to \mathbb{R}^m \qquad \pi(x) = \begin{pmatrix} \langle x, w_1 \rangle \\ \langle x, w_2 \rangle \\ \vdots \\ \langle x, w_m \rangle \end{pmatrix}$

PCA

Representation in PCA space

Conclusion:

We compute the eigenvectors and eigenvalues of the

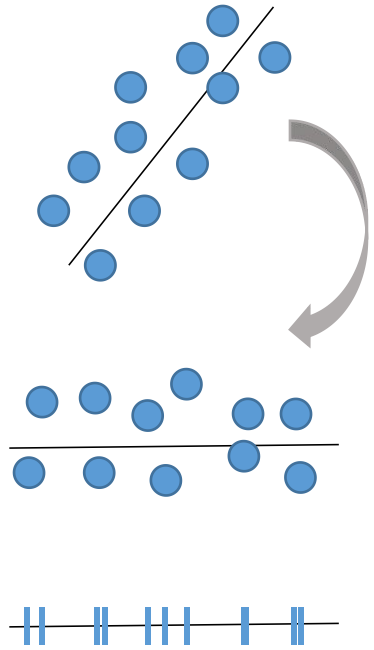
$$\text{Covariance matrix } C = \frac{1}{N}\, X \cdot X^T = \frac{1}{N} \sum_{i=1}^{N} x_i \cdot x_i^T \in \mathbb{R}^{n \times n}$$

Choose the first *m* eigenvectors corresponding to the *m* largest eigenvalues.

Then for all samples x from the distribution, we have

$$x \approx \sum_{j=1}^{m} \langle x, w_j \rangle w_j$$

PCA from 2d to 1d



Within the m-dimensional subspace $\text{span}\{w_j \mid j = 1, 2, \ldots, m\}$

x is encoded by the coefficients $\langle x, w_j \rangle$

$\Rightarrow$ The dimensionality reduction is defined by the representation $\pi : \mathbb{R}^n \to \mathbb{R}^m$

$$\pi(x) = \begin{pmatrix} \langle x, w_1 \rangle \\ \langle x, w_2 \rangle \\ \vdots \\ \langle x, w_m \rangle \end{pmatrix}$$

PCA

Representation in PCA space

Conclusion:

We compute the eigenvectors and eigenvalues of the

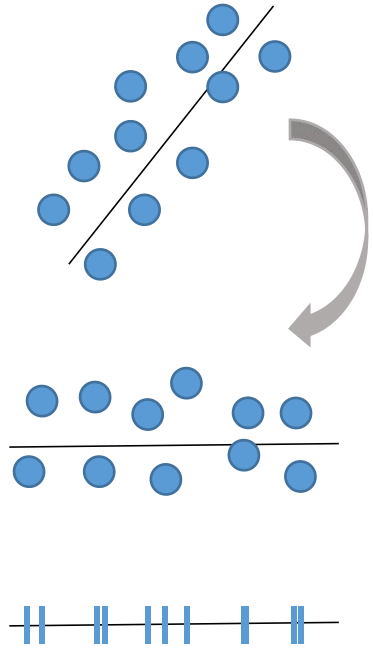
$$\text{Covariance matrix } C = \frac{1}{N}\, X \cdot X^T = \frac{1}{N} \sum_{i=1}^{N} x_i \cdot x_i^T \in \mathbb{R}^{n \times n}$$

Choose the first *m* eigenvectors corresponding to the *m* largest eigenvalues.

Then for all samples x from the distribution, we have

$$x \approx \sum_{j=1}^{m} \langle x, w_j \rangle w_j$$

PCA from 2d to 1d



Within the m-dimensional subspace $\text{span}\{w_j \mid j = 1, 2, \ldots, m\}$

x is encoded by the coefficients $\langle x, w_j \rangle$

Nice visualization at
* http://setosa.io/ev/principal-component-analysis/

$\Rightarrow$ The dimensionality reduction is defined by the representation $\pi : \mathbb{R}^n \to \mathbb{R}^m$

$$\pi(x) = \begin{pmatrix} \langle x, w_1 \rangle \\ \langle x, w_2 \rangle \\ \vdots \\ \langle x, w_m \rangle \end{pmatrix}$$

PCA

Computation from SVD

We can compute the PCA from the singular value decomposition on $X = [x_1|x_2|\dots|x_N] \in \mathbb{R}^{n\times N}$

$$X = W \cdot D \cdot V, \quad \text{where } W \in \mathbb{R}^{n\times n},\ D \in \mathbb{R}^{n\times N},\ V \in \mathbb{R}^{N\times N}$$

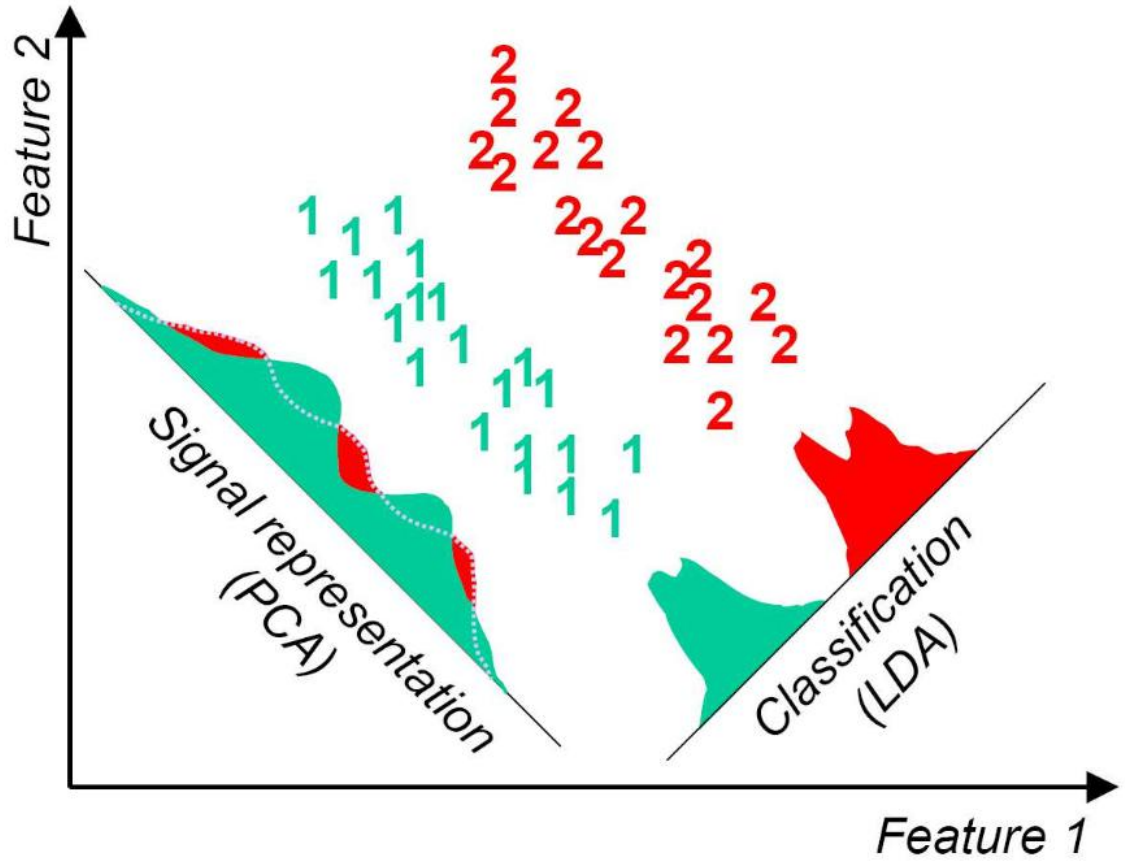
with W, V orthogonal matrices, and D a diagonal matrix.

W contains the eigenvectors of $C=XX^T$
D contains the square roots of the eigenvalues of C.

$$W^T X = \begin{pmatrix} \langle w_1, x_i \rangle \\ \langle w_2, x_i \rangle \\ \vdots \\ \langle w_n, x_i \rangle \end{pmatrix} = \pi(X)$$

PCA

vs Fisher's LDA

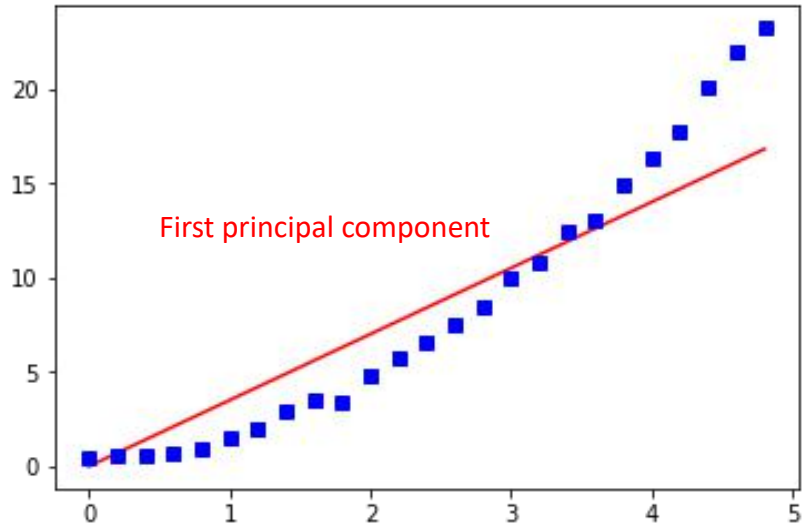


Nonlinear PCA

Idea

PCA is a **linear** method:

Principal components only give linear lines in the sample space



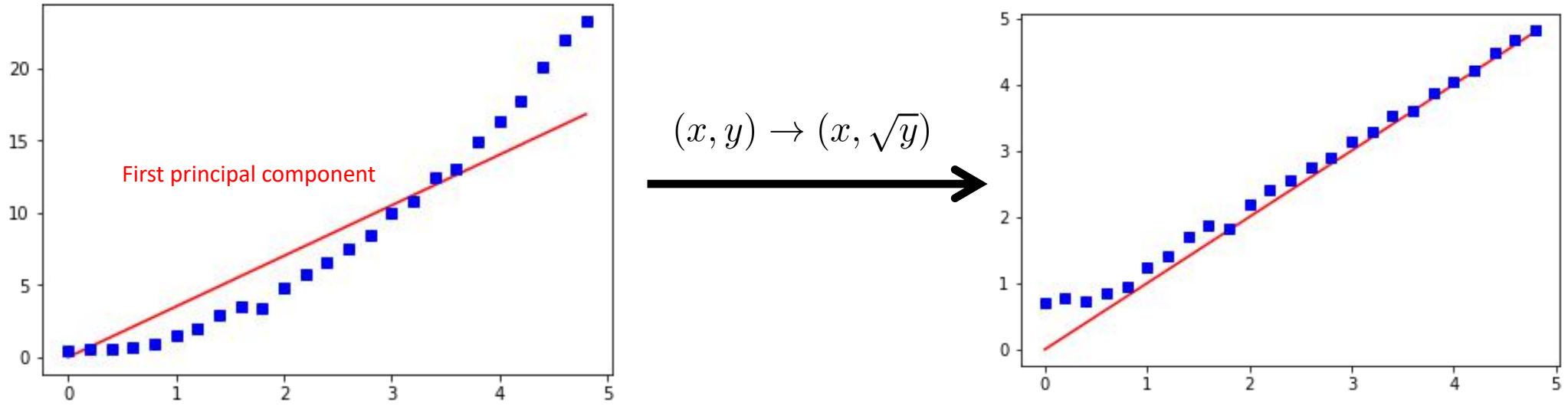
Nonlinear PCA

Idea

PCA is a **linear** method:

Principal components only give linear lines in the sample space

Similar to the idea with support vector machines, we may want to consider **non-linear features** and perform **linear PCA in feature space.**



Nonlinear PCA

Kernel PCA

As with Support Vector Machines, there is a **kernel trick** for PCA to turn linear PCA into **non-linear PCA** without the need to explicitly compute the feature space.

$$\text{Let } \phi(X) = [\phi(x_1)|\phi(x_2)|\dots|\phi(x_n)]$$

$$\text{The covariance matrix in feature space is given by } C^{\phi} = \frac{1}{n}\phi(X)\phi(X)^T = \frac{1}{n}\sum_{i=1}^{n}\phi(x_i)\phi(x_i)^T$$

$$\text{If } v \text{ is an eigenvector of } C^{\phi}, \text{ then } \lambda v = C^{\phi}v = \frac{1}{n}\phi(x_i)\phi(x_i)^T v$$

$$\Rightarrow v = \frac{1}{\lambda n}\sum_{i=1}^{n}(\phi(x_i)^T v)\phi(x_i) = \sum_{i=1}^{n}\alpha_i\phi(x_i) \text{ finding the principal component } v \text{ means finding the real numbers } \alpha_i$$

Nonlinear PCA

Kernel PCA

$$\text{Let } \phi(X) = [\phi(x_1)|\phi(x_2)|\dots|\phi(x_n)] \qquad K = \phi(X)^T\phi(X)$$

$$\alpha = \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_n \end{pmatrix} = \phi(X)^T v$$

$$C^\phi = \frac{1}{n}\phi(X)\phi(X)^T \qquad C^\phi v = \lambda v$$

Then

$$K\alpha = \phi(X)^T\phi(X)\phi(X)^T v = \phi(X)^T nCv = \phi(X)^T n\lambda v = n\lambda\alpha$$

Nonlinear PCA

Kernel PCA

$$\text{Let } \phi(X) = [\phi(x_1)|\phi(x_2)|\dots|\phi(x_n)] \qquad K = \phi(X)^T\phi(X)$$

$$C^\phi = \frac{1}{n}\phi(X)\phi(X)^T \qquad C^\phi v = \lambda v \qquad \alpha = \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_n \end{pmatrix} = \phi(X)^T v$$

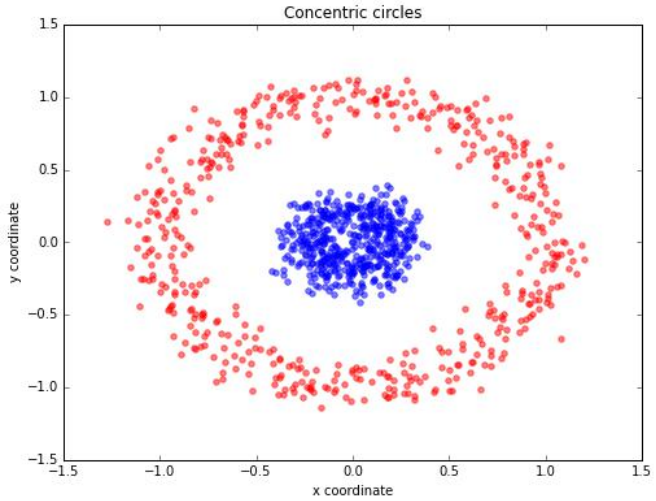
Then

$$K\alpha = \phi(X)^T\phi(X)\phi(X)^T v = \phi(X)^T nCv = \phi(X)^T n\lambda v = n\lambda\alpha$$

We have found an algorithm (find eigenvectors of K) phrased solely in terms of the kernel matrix K, so to solve the nonlinear PCA we do not need to compute the features explicitly, but it suffices to have K.

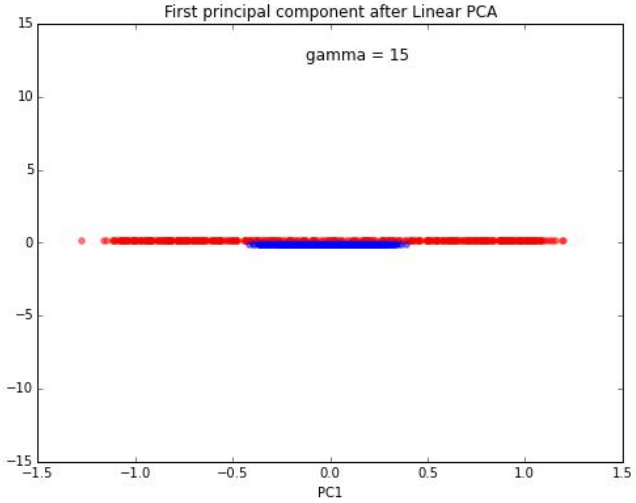
Nonlinear PCA

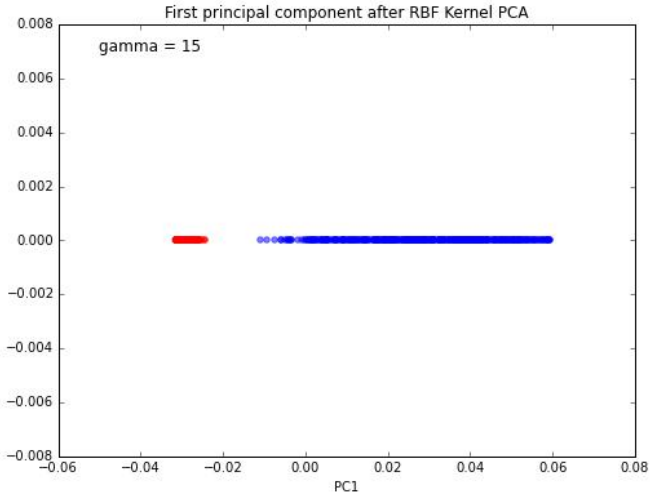
Kernel PCA



Linear PCA

Gaussian kernel PCA





For more examples, see
https://sebastianraschka.com/Articles/2014_kernel_pca.html

Dimensionality Reduction

Other Techniques

There exists

- Sparse PCA
- Robust PCA
- Self-organizing maps
- Independent Component Analysis (ICA)
- Kernel Entropy Analysis
- .....

„A survey of dimensionality reduction techniques“.
Sorzano, Vargas, Pascual-Montano
https://arxiv.org/ftp/arxiv/papers/1403/1403.2877.pdf

Dimensionality Reduction

Other Techniques

„A survey of dimensionality reduction techniques“.
Sorzano, Vargas, Pascual-Montano
https://arxiv.org/ftp/arxiv/papers/1403/1403.2877.pdf

There exists

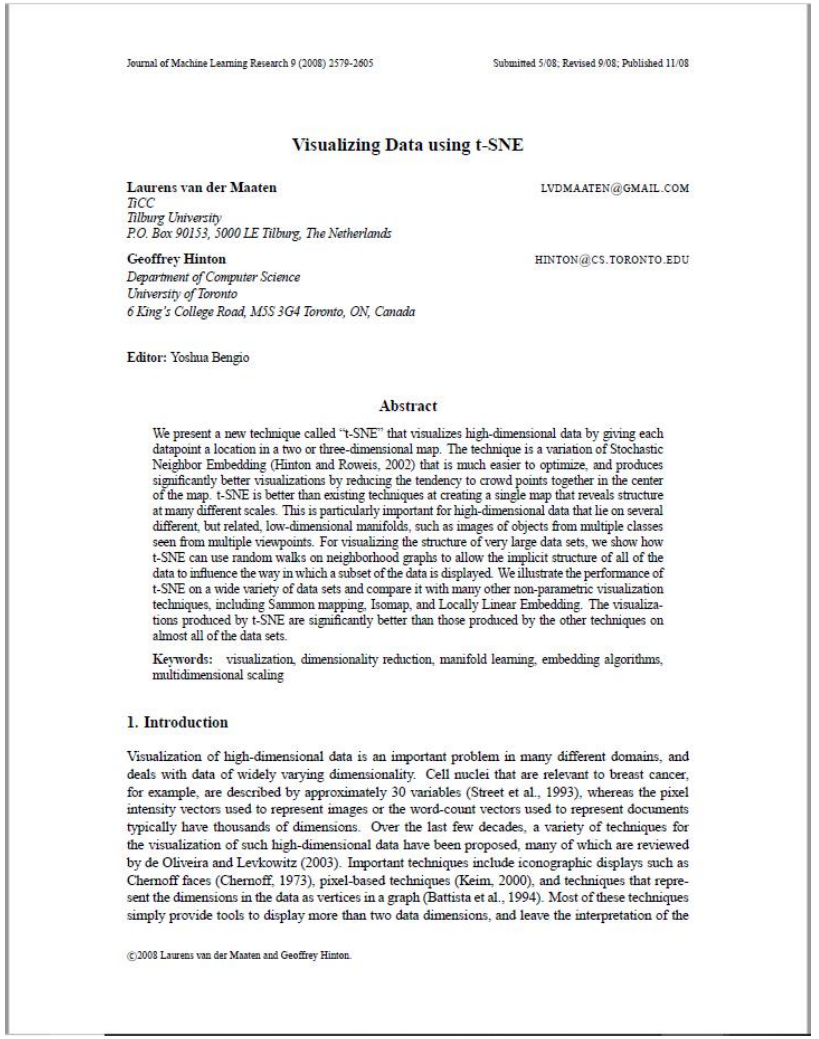
- Sparse PCA
- Robust PCA
- Self-organizing maps
- Independent Component Analysis (ICA)
- Kernel Entropy Analysis
- .....

We only treat one more method: t-SNE

Dimensionality Reduction

t-SNE

- New, powerful **non-linear dimensionality reduction** technique from 2008 (van der Maaten, Hinton)
- Uses **local relationships**, so **approximately preserves** clusters into lower dimension.
- Goal: Map into lower dimension such that nearest neighbor relation is approximately preserved (in probability)
- How it works:
  Models data in high dimension by a **probability distribution** and tries to **revocer this distribution in lower dimension**



Journal of Machine Learning Research 9 (2008) 2579-2605 Submitted 5/08; Revised 9/08; Published 11/08

Visualizing Data using t-SNE

Laurens van der Maaten LVDMAATEN@GMAIL.COM
TiCC
Tilburg University
P.O. Box 90153, 5000 LE Tilburg, The Netherlands

Geoffrey Hinton HINTON@CS.TORONTO.EDU
Department of Computer Science
University of Toronto
6 King's College Road, M5S 3G4 Toronto, ON, Canada

Editor: Yoshua Bengio

Abstract

We present a new technique called "t-SNE" that visualizes high-dimensional data by giving each datapoint a location in a two or three-dimensional map. The technique is a variation of Stochastic Neighbor Embedding (Hinton and Roweis, 2002) that is much easier to optimize, and produces significantly better visualizations by reducing the tendency to crowd points together in the center of the map. t-SNE is better than existing techniques at creating a single map that reveals structure at many different scales. This is particularly important for high-dimensional data that lie on several different, but related, low-dimensional manifolds, such as images of objects from multiple classes seen from multiple viewpoints. For visualizing the structure of very large data sets, we show how t-SNE can use random walks on neighborhood graphs to allow the implicit structure of all of the data to influence the way in which a subset of the data is displayed. We illustrate the performance of t-SNE on a wide variety of data sets and compare it with many other non-parametric visualization techniques, including Sammon mapping, Isomap, and Locally Linear Embedding. The visualizations produced by t-SNE are significantly better than those produced by the other techniques on almost all of the data sets.

Keywords: visualization, dimensionality reduction, manifold learning, embedding algorithms, multidimensional scaling

1. Introduction

Visualization of high-dimensional data is an important problem in many different domains, and deals with data of widely varying dimensionality. Cell nuclei that are relevant to breast cancer, for example, are described by approximately 30 variables (Street et al., 1993), whereas the pixel intensity vectors used to represent images or the word-count vectors used to represent documents typically have thousands of dimensions. Over the last few decades, a variety of techniques for the visualization of such high-dimensional data have been proposed, many of which are reviewed by de Oliveira and Levkowitz (2003). Important techniques include iconographic displays such as Chernoff faces (Chernoff, 1973), pixel-based techniques (Keim, 2000), and techniques that represent the dimensions in the data as vertices in a graph (Battista et al., 1994). Most of these techniques simply provide tools to display more than two data dimensions, and leave the interpretation of the

t-SNE

The S and the N

The S and the N in t-SNE stand for stochastic neighbor

Choose some x_i in the dataset. We model the probability that x_k ($k \neq i$) is considered a neighbor of x_i as

$$p_i(x_k) = \frac{\exp\left(\frac{-||x_i - x_k||^2}{2\sigma_i^2}\right)}{\sum_{j \neq i} \exp\left(\frac{-||x_i - x_j||^2}{2\sigma_i^2}\right)}$$

The parameters σ_i are defined by measure (called **perpexlity**) on the density of data around x_i, i.e. the number of neighbors close by.

t-SNE

The S and the E

The S and the E in t-SNE stand for stochastic neighbor

Choose some x_i in the dataset. We model the probability that x_k ,$k \neq i$, is considered a neighbor of x_i as

$$p_i(x_k) = \frac{\exp\left(\frac{-||x_i - x_k||^2}{2\sigma_i^2}\right)}{\sum_{j \neq i} \exp\left(\frac{-||x_i - x_j||^2}{2\sigma_i^2}\right)}$$

The parameters σ_i are defined by measure (called **perpexlity**) on the density of data around x_i, i.e. the number of neighbors close by.

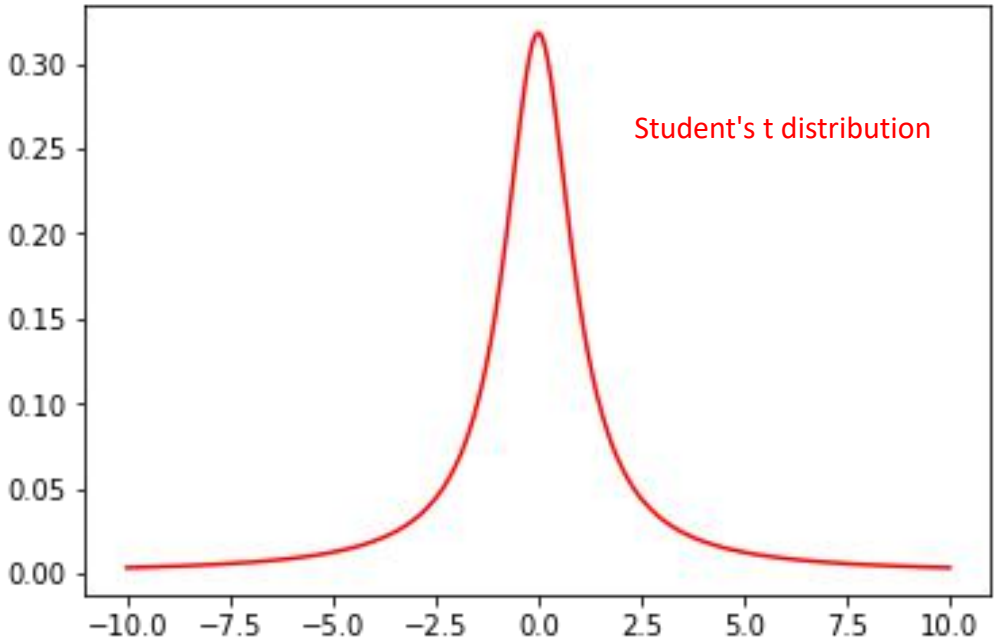
Now we aim to recover these probabilities in the lower dimensional space...
...well, almost these

t-SNE

The t

Instead of modelling the **probabilities of being neighbors** by Gaussians, **in the lower dimensional space** we will **model probabilities by the „Student's t-distribution“** (with one degre of freedom) with density

$$f(t) = \frac{1}{\pi(1+t^2)}$$

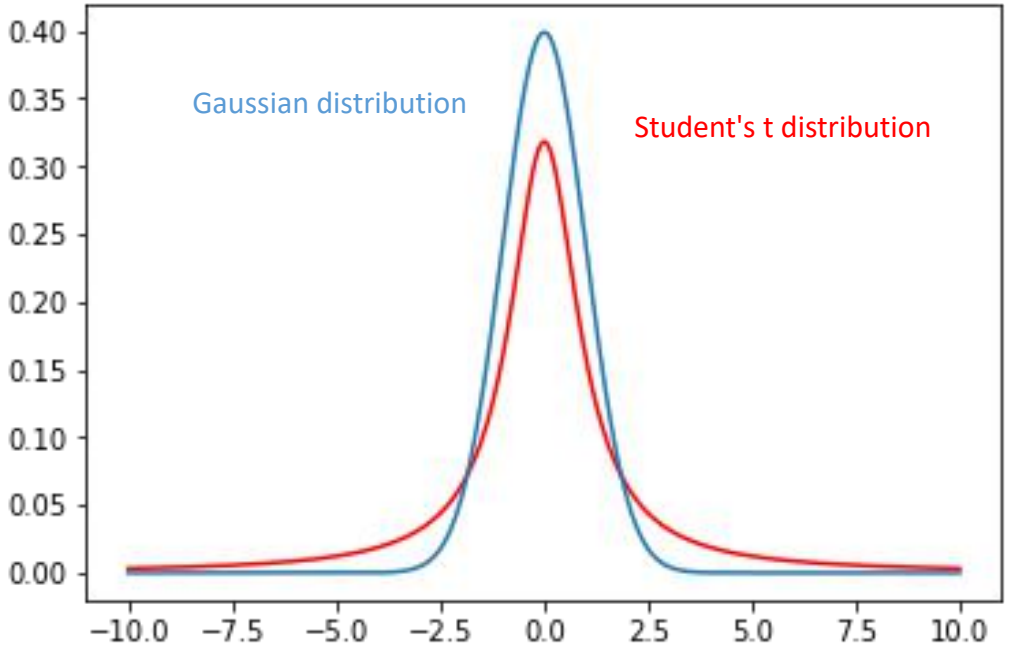


t-SNE

The t

Instead of modelling the **probabilities of being neighbors** by Gaussians, **in the lower dimensional space** we will **model probabilities by the „Student's t-distribution“** (with one degre of freedom) with density

$$f(t) = \frac{1}{\pi(1+t^2)}$$



Why this distribution and not Gaussians?
This is related to the so-called „**crowding problem**“ that points get very crowded in smaller dimensions. We want a larger tale to avoid this.

t-SNE

The t

Instead of modelling the **probabilities of being neighbors** by Gaussians, **in the lower dimensional space** we will **model probabilities by the „Student's t-distribution“** (with one degre of freedom) with density

$$f(t) = \frac{1}{\pi(1+t^2)}$$

With y_i denoting the lower dimensional representation of x_i, the probability of y_k being a neighbor of y_i is given by

$$q_i(y_k) = \frac{\frac{1}{1+||y_i - y_k||^2}}{\sum_{j \neq i} \frac{1}{1+||y_i - y_k||^2}}$$

t-SNE

The t

Instead of modelling the **probabilities of being neighbors** by Gaussians, **in the lower dimensional space** we will **model probabilities by the „Student's t-distribution“** (with one degre of freedom) with density

$$f(t) = \frac{1}{\pi(1+t^2)}$$

With y_i denoting the lower dimensional representation of x_i, the probability of y_k being a neighbor of y_i is given by

$$q_i(y_k) = \frac{\frac{1}{1+||y_i - y_k||^2}}{\sum_{j \neq i} \frac{1}{1+||y_i - y_k||^2}}$$

$$\downarrow$$

$$q_{ik}$$

Distribution in original high-dimensional space

$$p_i(x_k) = \frac{\exp\left(\frac{-||x_i - x_k||^2}{2\sigma_i^2}\right)}{\sum_{j \neq i} \exp\left(\frac{-||x_i - x_j||^2}{2\sigma_i^2}\right)}$$

$$\downarrow$$

$$p_{ik}$$

t-SNE

The t

Instead of modelling the **probabilities of being neighbors** by Gaussians, **in the lower dimensional space** we will **model probabilities by the „Student's t-distribution“** (with one degre of freedom) with density

$$f(t) = \frac{1}{\pi(1+t^2)}$$

With y_i denoting the lower dimensional representation of x_i, the probability of y_k being a neighbor of y_i is given by

$$q_i(y_k) = \frac{\frac{1}{1+||y_i - y_k||^2}}{\sum_{j \neq i} \frac{1}{1+||y_i - y_k||^2}}$$

Distribution in original high-dimensional space

$$p_i(x_k) = \frac{\exp\left(\frac{-||x_i - x_k||^2}{2\sigma_i^2}\right)}{\sum_{j \neq i} \exp\left(\frac{-||x_i - x_j||^2}{2\sigma_i^2}\right)}$$

q_{ik} ⟷ Distance measured as KL-divergence D_{KL} ⟷ p_{ik}

t-SNE

The t

Instead of modelling the **probabilities of being neighbors** by Gaussians, **in the lower dimensional space** we will **model probabilities by the „Student's t-distribution“** (with one degre of freedom) with density

$$f(t) = \frac{1}{\pi(1+t^2)}$$

With y_i denoting the lower dimensional representation of x_i, the probability of y_k being a neighbor of y_i is given by

$$q_i(y_k) = \frac{\frac{1}{1+||y_i - y_k||^2}}{\sum_{j \neq i} \frac{1}{1+||y_i - y_k||^2}}$$

Distribution in original high-dimensional space

$$p_i(x_k) = \frac{\exp\left(\frac{-||x_i - x_k||^2}{2\sigma_i^2}\right)}{\sum_{j \neq i} \exp\left(\frac{-||x_i - x_j||^2}{2\sigma_i^2}\right)}$$

q_{ik} ⟷ Distance measured as KL-divergence D_{KL} ⟷ p_{ik}

Minimize D_{KL} with Gradient Descent

t-SNE

The t

Instead of modelling the **probabilities of being neighbors** by Gaussians, **in the lower dimensional space** we will **model probabilities by the „Student's t-distribution“** (with one degre of freedom) with density

$$f(t) = \frac{1}{\pi(1+t^2)}$$

$$D_{KL}(P||Q) = \sum_{i,k} p_{ik} \log\left(\frac{p_{ik}}{q_{ik}}\right)$$

With y_i denoting the lower dimensional representation of x_i, the probability of y_k being a neighbor of y_i is given by

$$q_i(y_k) = \frac{\frac{1}{1+||y_i - y_k||^2}}{\sum_{j \neq i} \frac{1}{1+||y_i - y_k||^2}}$$

Distribution in original high-dimensional space

$$p_i(x_k) = \frac{\exp\left(\frac{-||x_i - x_k||^2}{2\sigma_i^2}\right)}{\sum_{j \neq i} \exp\left(\frac{-||x_i - x_j||^2}{2\sigma_i^2}\right)}$$

q_{ik}

Distance measured as **KL-divergence D_{KL}**

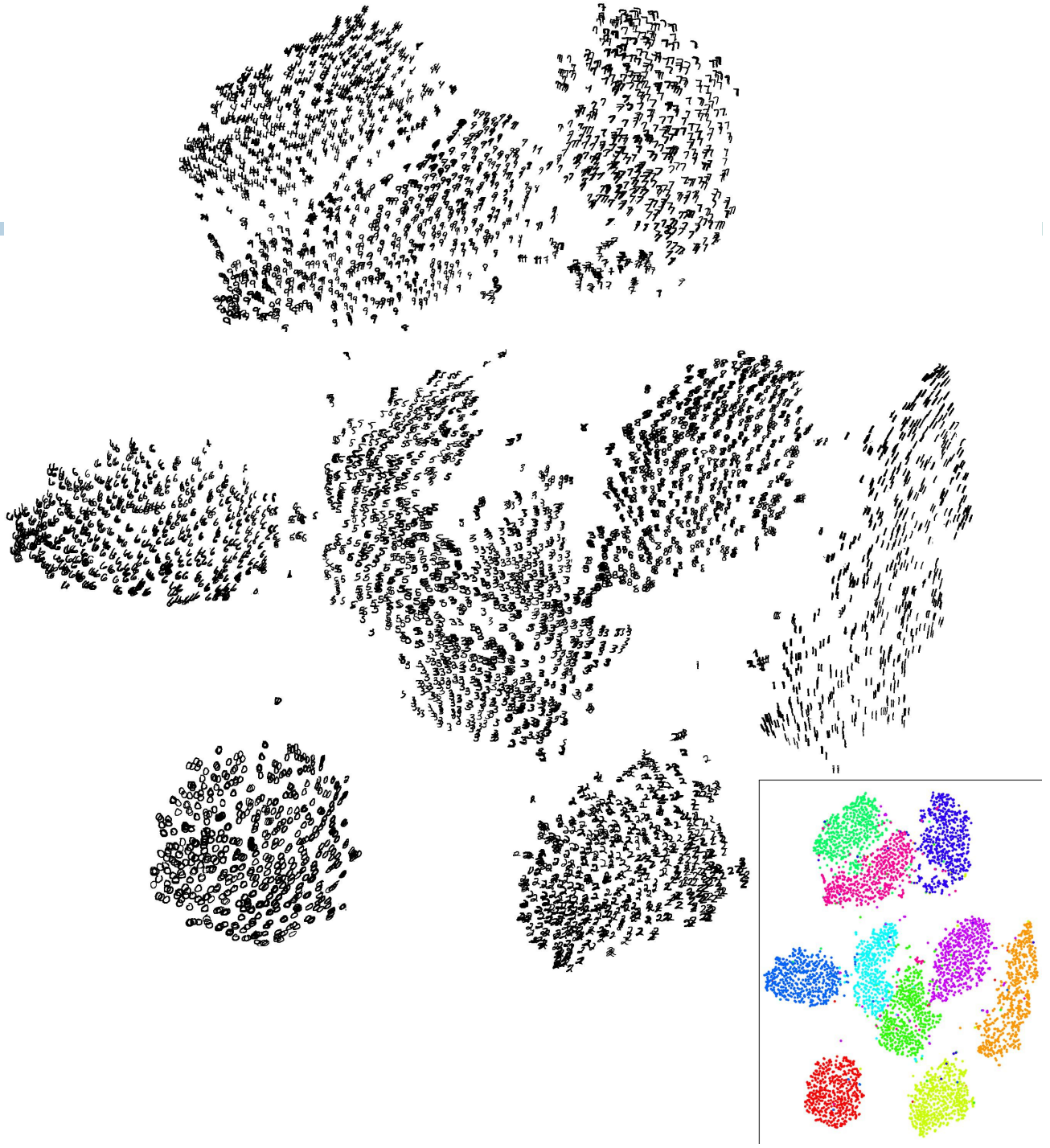
Minimize D_{KL} with Gradient Descent

p_{ik}

t-SNE

In action

Examples can be found on webpage of author's source code of t-SNE
https://lvdmaaten.github.io/tsne/



t-SNE

In action

Examples can be found on webpage of author's source code of t-SNE
https://lvdmaaten.github.io/tsne/

t-SNE

Properties

Pros

- Approximtely preserves **neighbors**
- Takes both **local and global information** into account
- Very **informative visualization**
- Can model **complex structures**

Cons

- We need to tune one hyperparameter (perplexity)
- Only find **local minima** with gradient descent
- Do **not** get a map, so **cannot display new points** in low-dimensional representation

t-SNE
In action

Nice visualization under https://distill.pub/2016/misread-tsne