

MADS-MMS – Mathematics and Multivariate Statistics

k Clustering

Prof. Dr. Stephan Doerfel



FACHHOCHSCHULE KIEL
University of Applied Sciences



Moodle (WiSe 24/25)

Agenda

Motivation

Basics

Construction of Central Points

- k -Means Algorithm

- Discussion of k -means

- Choosing a Good Cluster Number

Selecting Representative Instances

Outline

Motivation

Basics

Construction of Central Points

- k -Means Algorithm

- Discussion of k -means

- Choosing a Good Cluster Number

Selecting Representative Instances

Why k -Clustering?

- ▶ We have a fix number of groups that we want to split our data into.

Why k -Clustering?

- ▶ We have a fix number of groups that we want to split our data into.
- ▶ We have an intuition about the number of groups we want to split our data into.

Why k -Clustering?

- ▶ We have a fix number of groups that we want to split our data into.
- ▶ We have an intuition about the number of groups we want to split our data into.
- ▶ There are ways to determine “good” k , if we have none of the above.

Why k -Clustering?

- ▶ We have a fix number of groups that we want to split our data into.
- ▶ We have an intuition about the number of groups we want to split our data into.
- ▶ There are ways to determine “good” k , if we have none of the above.
- ▶ k clustering has simple algorithms, easy to implement and understand.

Chapter Goals

- ▶ understand the basic mechanics of well-known partitioning clustering algorithms (k -means and k -medoid)

Chapter Goals

- ▶ understand the basic mechanics of well-known partitioning clustering algorithms (k -means and k -medoid)
- ▶ understand the influence of the parameter k and methods for choosing it

Chapter Goals

- ▶ understand the basic mechanics of well-known partitioning clustering algorithms (k -means and k -medoid)
- ▶ understand the influence of the parameter k and methods for choosing it
- ▶ understand the influence of the initial clustering and approaches for mitigating it

Outline

Motivation

Basics

Construction of Central Points

- k -Means Algorithm

- Discussion of k -means

- Choosing a Good Cluster Number

Selecting Representative Instances

Basics

Task: Partition a dataset into k disjoint subsets.

Task: Partition a dataset into k disjoint subsets.

Definition 1 (Partition)

For a set D , $\mathcal{C} \subseteq 2^D$ is a partition (a clustering) if the following holds:

Task: Partition a dataset into k disjoint subsets.

Definition 1 (Partition)

For a set D , $\mathcal{C} \subseteq 2^D$ is a partition (a clustering) if the following holds:

- ▶ $\emptyset \notin \mathcal{C}$ (no cluster is empty)

Task: Partition a dataset into k disjoint subsets.

Definition 1 (Partition)

For a set D , $\mathcal{C} \subseteq 2^D$ is a partition (a clustering) if the following holds:

- ▶ $\emptyset \notin \mathcal{C}$ (no cluster is empty)
- ▶ $\bigcup_{A \in \mathcal{C}} A = D$ (each instance belongs to one cluster)

Task: Partition a dataset into k disjoint subsets.

Definition 1 (Partition)

For a set D , $\mathcal{C} \subseteq 2^D$ is a partition (a clustering) if the following holds:

- ▶ $\emptyset \notin \mathcal{C}$ (no cluster is empty)
- ▶ $\bigcup_{A \in \mathcal{C}} A = D$ (each instance belongs to one cluster)
- ▶ $\forall A, B \in \mathcal{C} : A \neq B \Rightarrow A \cap B = \emptyset$ (no overlap)

Task: Partition a dataset into k disjoint subsets.

Definition 1 (Partition)

For a set D , $\mathcal{C} \subseteq 2^D$ is a partition (a clustering) if the following holds:

- ▶ $\emptyset \notin \mathcal{C}$ (no cluster is empty)
- ▶ $\bigcup_{A \in \mathcal{C}} A = D$ (each instance belongs to one cluster)
- ▶ $\forall A, B \in \mathcal{C} : A \neq B \Rightarrow A \cap B = \emptyset$ (no overlap)

Definition 2

The set of all partitions of D is denoted by $\mathfrak{C}(D)$.

The set of all partitions of D of size k is denoted by $\mathfrak{C}_k(D)$.

Formal Definition

Definition 3 (Partitioning Clustering)

Let D be a set of data instances, $k \in \mathbb{N}$ and a cost function

$$\text{cost}: \mathfrak{C}_k(D) \rightarrow \mathbb{R}_{\geq 0}, \mathcal{C} \mapsto \text{cost}(\mathcal{C}).$$

Find a clustering $\mathcal{C}^{\text{opt}} \in \mathfrak{C}_k(D)$ that minimizes $\text{cost}(\mathcal{C}^{\text{opt}})$, i.e.

$$\mathcal{C}^{\text{opt}} = \arg \min_{\mathcal{C} \in \mathfrak{C}_k(D)} \text{cost}(\mathcal{C})$$

Formal Definition

Definition 3 (Partitioning Clustering)

Let D be a set of data instances, $k \in \mathbb{N}$ and a cost function

$$\text{cost}: \mathfrak{C}_k(D) \rightarrow \mathbb{R}_{\geq 0}, \mathcal{C} \mapsto \text{cost}(\mathcal{C}).$$

Find a clustering $\mathcal{C}^{\text{opt}} \in \mathfrak{C}_k(D)$ that minimizes $\text{cost}(\mathcal{C}^{\text{opt}})$, i.e.

$$\mathcal{C}^{\text{opt}} = \arg \min_{\mathcal{C} \in \mathfrak{C}_k(D)} \text{cost}(\mathcal{C})$$

Finding the optimal clustering \mathcal{C}^{opt} for given D, k , and cost is NP-complete. Hence: Find good approximations.

Solution: Local Optimization

Goal:

Solution: Local Optimization

Goal:

- ▶ partitioning into k clusters with approximately minimal costs

Solution: Local Optimization

Goal:

- ▶ partitioning into k clusters with approximately minimal costs

Locally Optimizing Method:

Solution: Local Optimization

Goal:

- ▶ partitioning into k clusters with approximately minimal costs

Locally Optimizing Method:

- ▶ choose k initial cluster representations

Solution: Local Optimization

Goal:

- ▶ partitioning into k clusters with approximately minimal costs

Locally Optimizing Method:

- ▶ choose k initial cluster representations
- ▶ optimize these representatives iteratively (lowering the costs)

Solution: Local Optimization

Goal:

- ▶ partitioning into k clusters with approximately minimal costs

Locally Optimizing Method:

- ▶ choose k initial cluster representations
- ▶ optimize these representatives iteratively (lowering the costs)
- ▶ stop when some given criterion is reached

Solution: Local Optimization

Goal:

- ▶ partitioning into k clusters with approximately minimal costs

Locally Optimizing Method:

- ▶ choose k initial cluster representations
- ▶ optimize these representatives iteratively (lowering the costs)
- ▶ stop when some given criterion is reached

Types of Cluster Representatives:

Solution: Local Optimization

Goal:

- ▶ partitioning into k clusters with approximately minimal costs

Locally Optimizing Method:

- ▶ choose k initial cluster representations
- ▶ optimize these representatives iteratively (lowering the costs)
- ▶ stop when some given criterion is reached

Types of Cluster Representatives:

- ▶ mean of a cluster's instances (construction of central points)

Solution: Local Optimization

Goal:

- ▶ partitioning into k clusters with approximately minimal costs

Locally Optimizing Method:

- ▶ choose k initial cluster representations
- ▶ optimize these representatives iteratively (lowering the costs)
- ▶ stop when some given criterion is reached

Types of Cluster Representatives:

- ▶ mean of a cluster's instances (construction of central points)
- ▶ element of cluster (selection of representative data instances)

Solution: Local Optimization

Goal:

- ▶ partitioning into k clusters with approximately minimal costs

Locally Optimizing Method:

- ▶ choose k initial cluster representations
- ▶ optimize these representatives iteratively (lowering the costs)
- ▶ stop when some given criterion is reached

Types of Cluster Representatives:

- ▶ mean of a cluster's instances (construction of central points)
- ▶ element of cluster (selection of representative data instances)
- ▶ (cluster probability distribution (maximizing expectation) ⌚)

Outline

Motivation

Basics

Construction of Central Points

- k -Means Algorithm

- Discussion of k -means

- Choosing a Good Cluster Number

Selecting Representative Instances

Agenda

Motivation

Basics

Construction of Central Points

k -Means Algorithm

Discussion of k -means

Choosing a Good Cluster Number

Selecting Representative Instances

Idea

Assumption

data is grouped around a fix number (k) of central points in the data space (not data set!) – each such point represents one cluster

Idea

Assumption

data is grouped around a fix number (k) of central points in the data space (not data set!) – each such point represents one cluster

Approach

Idea

Assumption

data is grouped around a fix number (k) of central points in the data space (not data set!) – each such point represents one cluster

Approach

- ▶ try to determine those k central points, called **centroids**

Idea

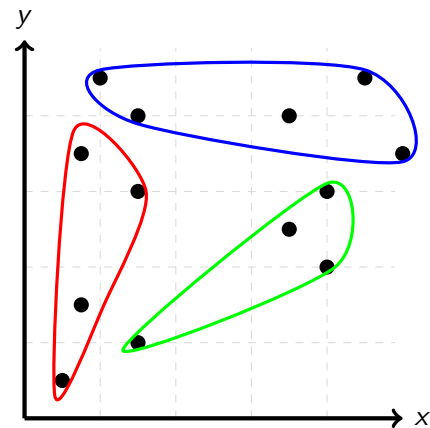
Assumption

data is grouped around a fix number (k) of central points in the data space (not data set!) – each such point represents one cluster

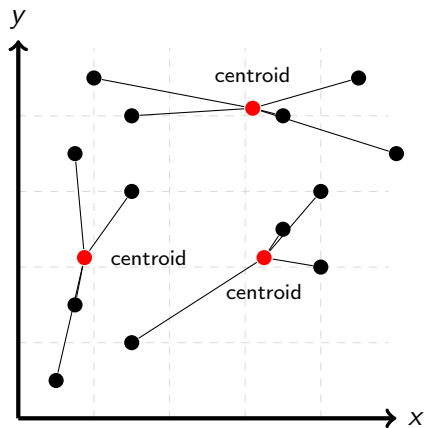
Approach

- ▶ try to determine those k central points, called **centroids**
- ▶ each data instance is assigned the closests of the central points

Examples, Bad/Good Clusterings, $k = 3$

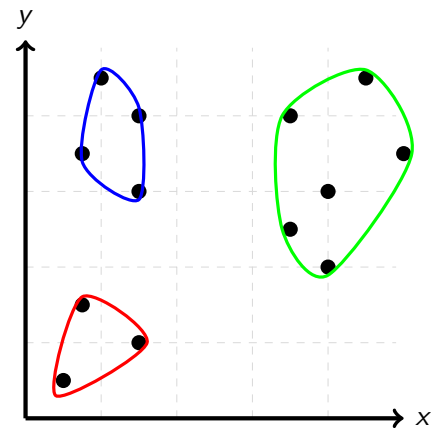


Cluster

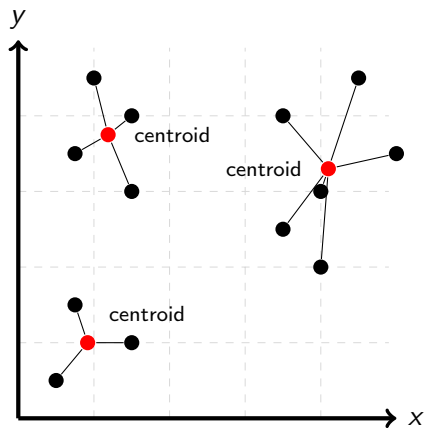


Cluster-Centroids

Examples, Bad/Good Clusterings, $k = 3$



Cluster



Cluster-Centroids

Construction of Central Points

Instances: vectors $p = (x_{p_1}, \dots, x_{p_d})$ in a Euclidean vector space

Construction of Central Points

Instances: vectors $p = (x_{p_1}, \dots, x_{p_d})$ in a Euclidean vector space

Distance: Euclidean distance

Construction of Central Points

Instances: vectors $p = (x_{p_1}, \dots, x_{p_d})$ in a Euclidean vector space

Distance: Euclidean distance

Central Point: **Centroid** $\mu_C :=$ mean of the vectors in Cluster C

Construction of Central Points

Instances: vectors $p = (x_{p_1}, \dots, x_{p_d})$ in a Euclidean vector space

Distance: Euclidean distance

Central Point: **Centroid** $\mu_C :=$ mean of the vectors in Cluster C

Cluster-Cost: measure for the (non-)compactness of a cluster C –

inertia:

$$\text{cost}(C) = TD^2(C) := \sum_{p \in C} \text{dist}(p, \mu_C)^2$$

Construction of Central Points

Instances: vectors $p = (x_{p_1}, \dots, x_{p_d})$ in a Euclidean vector space

Distance: Euclidean distance

Central Point: **Centroid** $\mu_C :=$ mean of the vectors in Cluster C

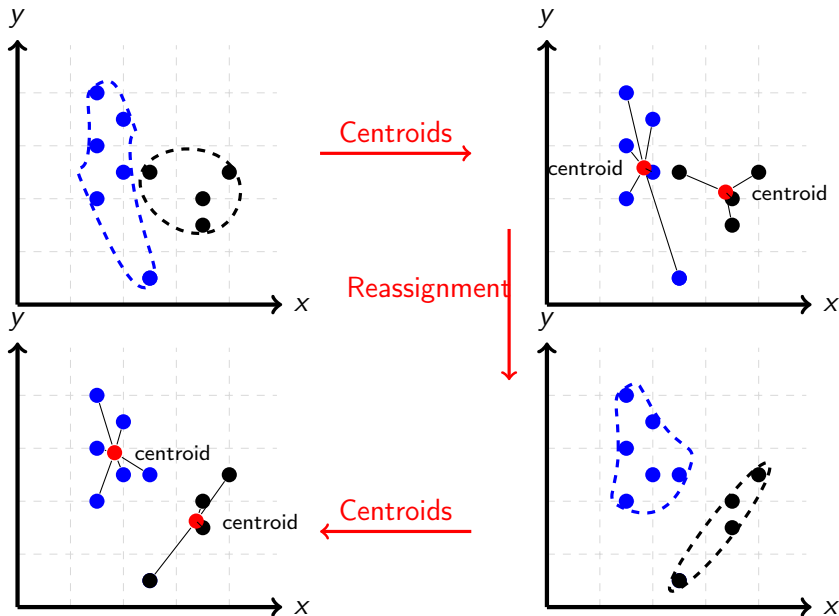
Cluster-Cost: measure for the (non-)compactness of a cluster C –
inertia:

$$\text{cost}(C) = TD^2(C) := \sum_{p \in C} \text{dist}(p, \mu_C)^2$$

Clustering-Cost: measure for the (non-)compactness of a clustering \mathcal{C} :

$$\text{cost}(\mathcal{C}) = TD^2(\mathcal{C}) := \sum_{C \in \mathcal{C}} TD^2(C)$$

Base-Algorithm



Base Algorithm – Variation Minimization

Initialization

Base Algorithm – Variation Minimization

Initialization

1. choose k data instances from the dataset as centroids

Base Algorithm – Variation Minimization

Initialization

1. choose k data instances from the dataset as centroids
2. assign each data instance to the closest centroid, thus create initial clustering \mathcal{C}

Base Algorithm – Variation Minimization

Initialization

1. choose k data instances from the dataset as centroids
2. assign each data instance to the closest centroid, thus create initial clustering \mathcal{C}
3. compute clustering inertia cost(\mathcal{C})

Base Algorithm – Variation Minimization

Initialization

1. choose k data instances from the dataset as centroids
2. assign each data instance to the closest centroid, thus create initial clustering \mathcal{C}
3. compute clustering inertia cost(\mathcal{C})

Iteration

Base Algorithm – Variation Minimization

Initialization

1. choose k data instances from the dataset as centroids
2. assign each data instance to the closest centroid, thus create initial clustering \mathcal{C}
3. compute clustering inertia cost(\mathcal{C})

Iteration

1. re-compute centroids for \mathcal{C}

Base Algorithm – Variation Minimization

Initialization

1. choose k data instances from the dataset as centroids
2. assign each data instance to the closest centroid, thus create initial clustering \mathcal{C}
3. compute clustering inertia cost(\mathcal{C})

Iteration

1. re-compute centroids for \mathcal{C}
2. assign each data instance to the closest centroid, thus create a new clustering \mathcal{C}'

Base Algorithm – Variation Minimization

Initialization

1. choose k data instances from the dataset as centroids
2. assign each data instance to the closest centroid, thus create initial clustering \mathcal{C}
3. compute clustering inertia cost(\mathcal{C})

Iteration

1. re-compute centroids for \mathcal{C}
2. assign each data instance to the closest centroid, thus create a new clustering \mathcal{C}'
3. compute clustering inertia cost(\mathcal{C}')

Base Algorithm – Variation Minimization

Initialization

1. choose k data instances from the dataset as centroids
2. assign each data instance to the closest centroid, thus create initial clustering \mathcal{C}
3. compute clustering inertia cost(\mathcal{C})

Iteration

1. re-compute centroids for \mathcal{C}
2. assign each data instance to the closest centroid, thus create a new clustering \mathcal{C}'
3. compute clustering inertia cost(\mathcal{C}')
4. repeat 1 through 3 with $\mathcal{C} := \mathcal{C}'$ until reaching some stop-criterion

Base Algorithm – Variation Minimization

Initialization

1. choose k data instances from the dataset as centroids
2. assign each data instance to the closest centroid, thus create initial clustering \mathcal{C}
3. compute clustering inertia cost(\mathcal{C})

Iteration

1. re-compute centroids for \mathcal{C}
2. assign each data instance to the closest centroid, thus create a new clustering \mathcal{C}'
3. compute clustering inertia cost(\mathcal{C}')
4. repeat 1 through 3 with $\mathcal{C} := \mathcal{C}'$ until reaching some stop-criterion

Stopping Criteria

Base Algorithm – Variation Minimization

Initialization

1. choose k data instances from the dataset as centroids
2. assign each data instance to the closest centroid, thus create initial clustering \mathcal{C}
3. compute clustering inertia cost(\mathcal{C})

Iteration

1. re-compute centroids for \mathcal{C}
2. assign each data instance to the closest centroid, thus create a new clustering \mathcal{C}'
3. compute clustering inertia cost(\mathcal{C}')
4. repeat 1 through 3 with $\mathcal{C} := \mathcal{C}'$ until reaching some stop-criterion

Stopping Criteria

- $\mathcal{C} = \mathcal{C}'$

Base Algorithm – Variation Minimization

Initialization

1. choose k data instances from the dataset as centroids
2. assign each data instance to the closest centroid, thus create initial clustering \mathcal{C}
3. compute clustering inertia cost(\mathcal{C})

Iteration

1. re-compute centroids for \mathcal{C}
2. assign each data instance to the closest centroid, thus create a new clustering \mathcal{C}'
3. compute clustering inertia cost(\mathcal{C}')
4. repeat 1 through 3 with $\mathcal{C} := \mathcal{C}'$ until reaching some stop-criterion

Stopping Criteria

- ▶ $\mathcal{C} = \mathcal{C}'$
- ▶ $|\text{cost}(\mathcal{C}) - \text{cost}(\mathcal{C}')|$ below some threshold

Different Variations of the Base-Algorithm

k-means:

Different Variations of the Base-Algorithm

k-means:

- ▶ similar, but instead of recomputing the full clustering, find one instance that should belong to a closer centroid and immediately recompute the centroids

Different Variations of the Base-Algorithm

k-means:

- ▶ similar, but instead of recomputing the full clustering, find one instance that should belong to a closer centroid and immediately recompute the centroids
- ▶ *k*-means mainly has the properties of the base algorithm

Different Variations of the Base-Algorithm

k-means:

- ▶ similar, but instead of recomputing the full clustering, find one instance that should belong to a closer centroid and immediately recompute the centroids
- ▶ *k*-means mainly has the properties of the base algorithm
- ▶ *k*-means depends on the order of the dataset

Different Variations of the Base-Algorithm

k-means:

- ▶ similar, but instead of recomputing the full clustering, find one instance that should belong to a closer centroid and immediately recompute the centroids
- ▶ *k*-means mainly has the properties of the base algorithm
- ▶ *k*-means depends on the order of the dataset
- ▶ computationally less expensive (easy centroid updates instead of full recomputation)

Different Variations of the Base-Algorithm

k-means:

- ▶ similar, but instead of recomputing the full clustering, find one instance that should belong to a closer centroid and immediately recompute the centroids
- ▶ *k*-means mainly has the properties of the base algorithm
- ▶ *k*-means depends on the order of the dataset
- ▶ computationally less expensive (easy centroid updates instead of full recomputation)
- ▶ faster convergence

Different Variations of the Base-Algorithm

k-means:

- ▶ similar, but instead of recomputing the full clustering, find one instance that should belong to a closer centroid and immediately recompute the centroids
- ▶ *k*-means mainly has the properties of the base algorithm
- ▶ *k*-means depends on the order of the dataset
- ▶ computationally less expensive (easy centroid updates instead of full recomputation)
- ▶ faster convergence

k-means++:

Different Variations of the Base-Algorithm

k-means:

- ▶ similar, but instead of recomputing the full clustering, find one instance that should belong to a closer centroid and immediately recompute the centroids
- ▶ *k*-means mainly has the properties of the base algorithm
- ▶ *k*-means depends on the order of the dataset
- ▶ computationally less expensive (easy centroid updates instead of full recomputation)
- ▶ faster convergence

k-means++:

- ▶ same as *k*-means but with sophisticated initialization

Different Variations of the Base-Algorithm

k-means:

- ▶ similar, but instead of recomputing the full clustering, find one instance that should belong to a closer centroid and immediately recompute the centroids
- ▶ *k*-means mainly has the properties of the base algorithm
- ▶ *k*-means depends on the order of the dataset
- ▶ computationally less expensive (easy centroid updates instead of full recomputation)
- ▶ faster convergence

k-means++:

- ▶ same as *k*-means but with sophisticated initialization
- ▶ initial centroids are chosen iteratively

Different Variations of the Base-Algorithm

k-means:

- ▶ similar, but instead of recomputing the full clustering, find one instance that should belong to a closer centroid and immediately recompute the centroids
- ▶ *k*-means mainly has the properties of the base algorithm
- ▶ *k*-means depends on the order of the dataset
- ▶ computationally less expensive (easy centroid updates instead of full recomputation)
- ▶ faster convergence

k-means++:

- ▶ same as *k*-means but with sophisticated initialization
- ▶ initial centroids are chosen iteratively
 - ▶ first centroid is chosen from the dataset at random

Different Variations of the Base-Algorithm

k-means:

- ▶ similar, but instead of recomputing the full clustering, find one instance that should belong to a closer centroid and immediately recompute the centroids
- ▶ *k*-means mainly has the properties of the base algorithm
- ▶ *k*-means depends on the order of the dataset
- ▶ computationally less expensive (easy centroid updates instead of full recomputation)
- ▶ faster convergence

k-means++:

- ▶ same as *k*-means but with sophisticated initialization
- ▶ initial centroids are chosen iteratively
 - ▶ first centroid is chosen from the dataset at random
 - ▶ each further centroid: chosen from dataset with probability proportional to the squared distance to its nearest centroid

Different Variations of the Base-Algorithm

k-means:

- ▶ similar, but instead of recomputing the full clustering, find one instance that should belong to a closer centroid and immediately recompute the centroids
- ▶ *k*-means mainly has the properties of the base algorithm
- ▶ *k*-means depends on the order of the dataset
- ▶ computationally less expensive (easy centroid updates instead of full recomputation)
- ▶ faster convergence

k-means++:

- ▶ same as *k*-means but with sophisticated initialization
- ▶ initial centroids are chosen iteratively
 - ▶ first centroid is chosen from the dataset at random
 - ▶ each further centroid: chosen from dataset with probability proportional to the squared distance to its nearest centroid
- ▶ longer initialization, speed-up of convergence, decrease of squared distance

k-means in Python

`sklearn.cluster.KMeans`

- ▶ default: *k*-means++

k-means in Python

`sklearn.cluster.KMeans`

- ▶ default: *k*-means++
- ▶ setup to run multiple times (`n_iter`) with different initialization, returns the best clustering

k-means in Python

`sklearn.cluster.KMeans`

- ▶ default: *k*-means++
- ▶ setup to run multiple times (`n_iter`) with different initialization, returns the best clustering
- ▶ abort criteria

k-means in Python

`sklearn.cluster.KMeans`

- ▶ default: *k*-means++
- ▶ setup to run multiple times (`n_iter`) with different initialization, returns the best clustering
- ▶ abort criteria
 - ▶ `max_iter`: maximum number of iterations

k-means in Python

`sklearn.cluster.KMeans`

- ▶ default: *k*-means++
- ▶ setup to run multiple times (`n_iter`) with different initialization, returns the best clustering
- ▶ abort criteria
 - ▶ `max_iter`: maximum number of iterations
 - ▶ `tol`: threshold on the change in the cluster centers between two iterations to assume convergence

k-means in Python

`sklearn.cluster.KMeans`

- ▶ default: *k*-means++
- ▶ setup to run multiple times (`n_iter`) with different initialization, returns the best clustering
- ▶ abort criteria
 - ▶ `max_iter`: maximum number of iterations
 - ▶ `tol`: threshold on the change in the cluster centers between two iterations to assume convergence

 Notebook 06_1_k_means_synthetic

Agenda

Motivation

Basics

Construction of Central Points

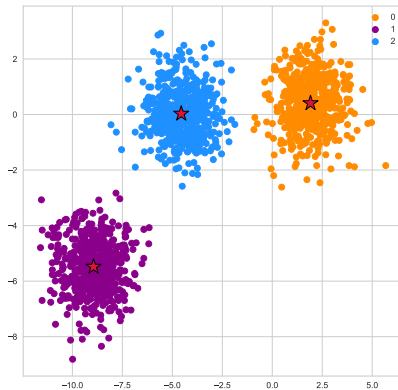
k-Means Algorithm

Discussion of *k*-means

Choosing a Good Cluster Number

Selecting Representative Instances

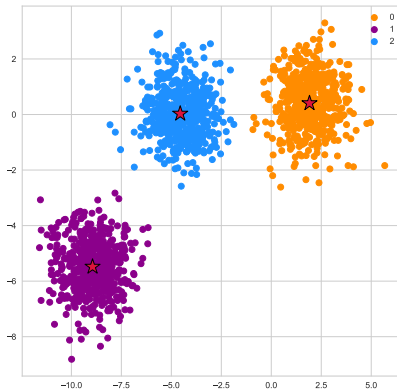
k -Means Example¹



► $k = 3$

¹adapted from an [!\[\]\(e5b174fc79657ab6e1a83733a09e0656_img.jpg\) sklearn tutorial](#)

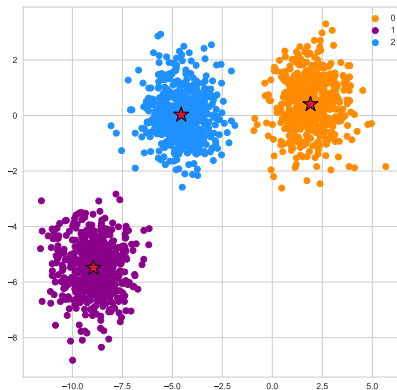
k -Means Example¹



- $k = 3$
- stars mark the centroids

¹adapted from an  sklearn tutorial

k -Means Example¹



- ▶ $k = 3$
- ▶ stars mark the centroids
- ▶ in 2D easy plausibility check

¹adapted from an  sklearn tutorial

Discussion of k -means

positive:

Discussion of k -means

positive:

- ▶ efficiency:

Discussion of k -means

positive:

- ▶ efficiency:
 - ▶ **Computational Effort:** $O(k \cdot n)$ per iteration,

Discussion of k -means

positive:

- ▶ efficiency:
 - ▶ **Computational Effort:** $O(k \cdot n)$ per iteration,
 - ▶ number of required iterations usually small (≈ 5 to 10).

Discussion of k -means

positive:

- ▶ efficiency:
 - ▶ **Computational Effort:** $O(k \cdot n)$ per iteration,
 - ▶ number of required iterations usually small (≈ 5 to 10).
- ▶ simple implementation
 - ➔ most popular partitioning clustering approach

Discussion of k -means

positive:

- ▶ efficiency:
 - ▶ **Computational Effort:** $O(k \cdot n)$ per iteration,
 - ▶ number of required iterations usually small (≈ 5 to 10).
- ▶ simple implementation
 - most popular partitioning clustering approach
- ▶ straightforward → easily applicable, easily explained

Discussion of k -means

positive:

- ▶ efficiency:
 - ▶ **Computational Effort:** $O(k \cdot n)$ per iteration,
 - ▶ number of required iterations usually small (≈ 5 to 10).
- ▶ simple implementation
 - most popular partitioning clustering approach
- ▶ straightforward → easily applicable, easily explained

negative:

Discussion of k -means

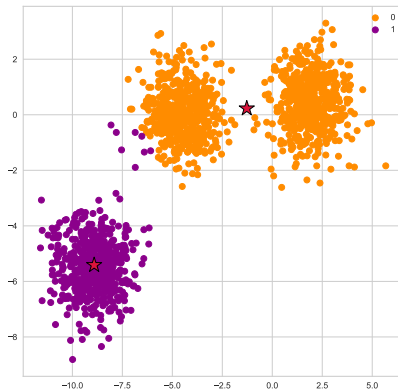
positive:

- ▶ efficiency:
 - ▶ **Computational Effort:** $O(k \cdot n)$ per iteration,
 - ▶ number of required iterations usually small (≈ 5 to 10).
- ▶ simple implementation
 - most popular partitioning clustering approach
- ▶ straightforward → easily applicable, easily explained

negative:

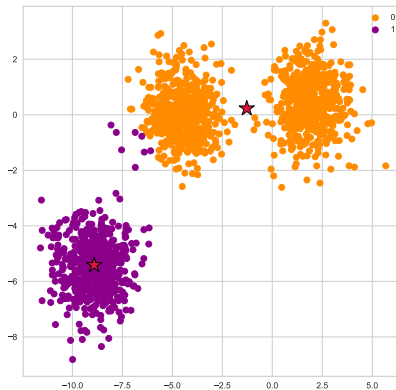
- ▶ let's look at a couple of examples

The k of k -Means²



²adapted from an  sklearn tutorial

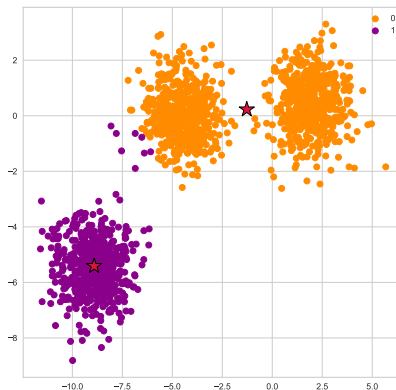
The k of k -Means²



- we need the “correct” k , upfront

²adapted from an  sklearn tutorial

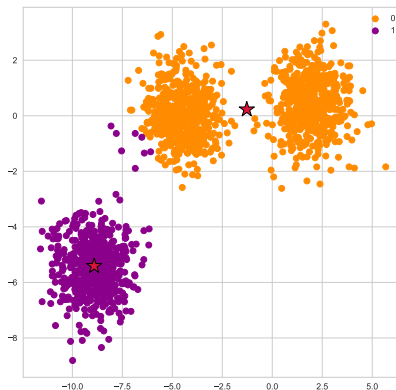
The k of k -Means²



- we need the “correct” k , upfront
- sometimes easy to see in a plot

²adapted from an  sklearn tutorial

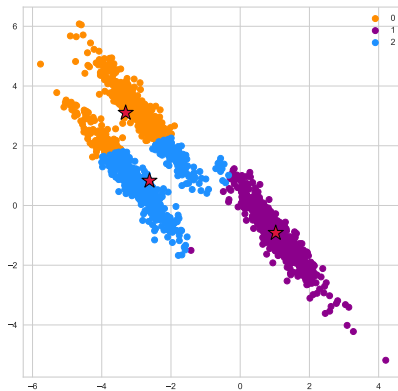
The k of k -Means²



- ▶ we need the “correct” k , upfront
- ▶ sometimes easy to see in a plot
- ▶ much more difficult for multidimensional data

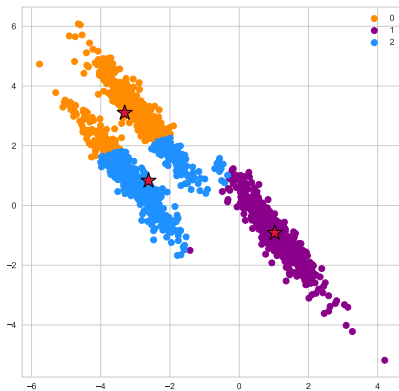
²adapted from an  sklearn tutorial

The Form of k -Means Clusters³



³adapted from an  sklearn tutorial

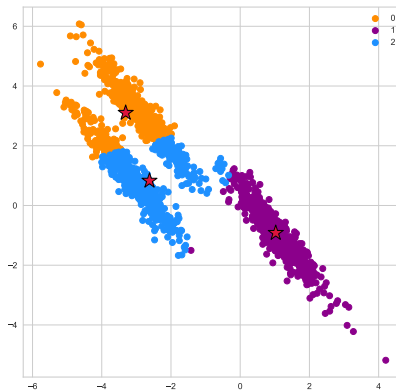
The Form of k -Means Clusters³



- k -means clusters are circular surroundings of the centroids

³adapted from an  sklearn tutorial

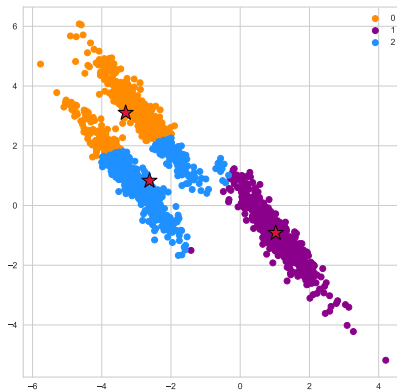
The Form of k -Means Clusters³



- ▶ k -means clusters are circular surroundings of the centroids
- ▶ the centroids induce a Voronoi diagram

³adapted from an  sklearn tutorial

The Form of k -Means Clusters³



- ▶ k -means clusters are circular surroundings of the centroids
- ▶ the centroids induce a Voronoi diagram
- ▶ the form does not necessarily reflect the real data structure

³adapted from an  sklearn tutorial

Discussion of k -means

negative:

Discussion of k -means

negative:

- ▶ choosing k properly is often difficult → ⌚ silhouette coefficient

Discussion of k -means

negative:

- ▶ choosing k properly is often difficult → ⌚ silhouette coefficient
- ▶ “real” clusters must have convex form for a good fit → ⌚ density-based clustering

Discussion of k -means

negative:

- ▶ choosing k properly is often difficult → ⚠ silhouette coefficient
- ▶ “real” clusters must have convex form for a good fit → ⚠ density-based clustering
- ▶ sensitive to noise and outliers (all instances enter the computation of centroids) →

Discussion of k -means

negative:

- ▶ choosing k properly is often difficult → ⌚ silhouette coefficient
- ▶ “real” clusters must have convex form for a good fit → ⌚ density-based clustering
- ▶ sensitive to noise and outliers (all instances enter the computation of centroids) →
 - ▶ drop outliers before running the algorithm

Discussion of k -means

negative:

- ▶ choosing k properly is often difficult → ⌚ silhouette coefficient
- ▶ “real” clusters must have convex form for a good fit → ⌚ density-based clustering
- ▶ sensitive to noise and outliers (all instances enter the computation of centroids) →
 - ▶ drop outliers before running the algorithm
 - ▶ ⌚ density-based clustering

Discussion of k -means

negative:

- ▶ choosing k properly is often difficult → ⌚ silhouette coefficient
- ▶ “real” clusters must have convex form for a good fit → ⌚ density-based clustering
- ▶ sensitive to noise and outliers (all instances enter the computation of centroids) →
 - ▶ drop outliers before running the algorithm
 - ▶ ⌚ density-based clustering
- ▶ result strongly depends on initial choice of centroids → run multiple times, use best result

Discussion of k -means

negative:

- ▶ choosing k properly is often difficult → ⚠ silhouette coefficient
- ▶ “real” clusters must have convex form for a good fit → ⚠ density-based clustering
- ▶ sensitive to noise and outliers (all instances enter the computation of centroids) →
 - ▶ drop outliers before running the algorithm
 - ▶ ⚠ density-based clustering
- ▶ result strongly depends on initial choice of centroids → run multiple times, use best result
- ▶ no confidence for cluster memberships → ⚠ probabilistic clustering

Agenda

Motivation

Basics

Construction of Central Points

k-Means Algorithm

Discussion of *k*-means

Choosing a Good Cluster Number

Selecting Representative Instances

Choosing k

method:

- ▶ create a clustering for each $k = 2, \dots, n - 1$
- ▶ choose k with the best clustering – according to a quality measure

measure for a clustering's quality:

- ▶ needs to be independent of k
- ▶ TD^2 decreases monotonously with increasing k
- ▶ TD^2 is thus unsuitable as quality measure
- ▶ similar effect for k -Medoid and EM ⌚

Choosing k – The Silhouette Coefficient

Definition 4 (Silhouette Coefficient)

For a given clustering \mathcal{C} on a dataset D , the **silhouette** $s(o)$ of an instance $o \in D$ is given as $s(o) = 0$ if o 's cluster has only the one element, and otherwise

$$s(o) = \frac{b(o) - a(o)}{\max\{a(o), b(o)\}}$$


where

- ▶ $a(o)$ is the mean distance to the other elements in o 's cluster, and
- ▶ $b(o)$ the mean distance to the elements of the “nearest” cluster.

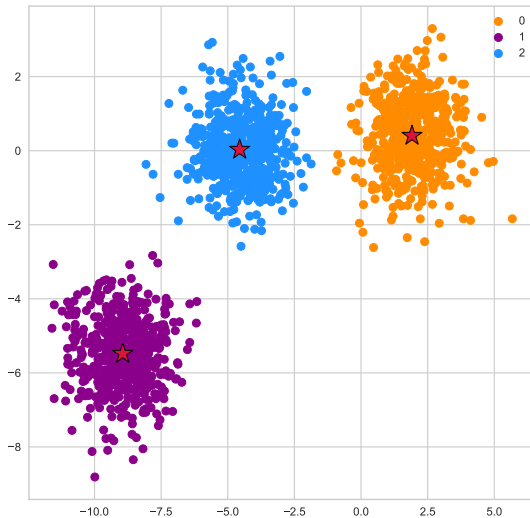
The nearest cluster is the one with the lowest mean distance to o . The silhouette coefficient of a clustering is the mean of the objects' silhouettes.

Properties of the Silhouette Coefficient

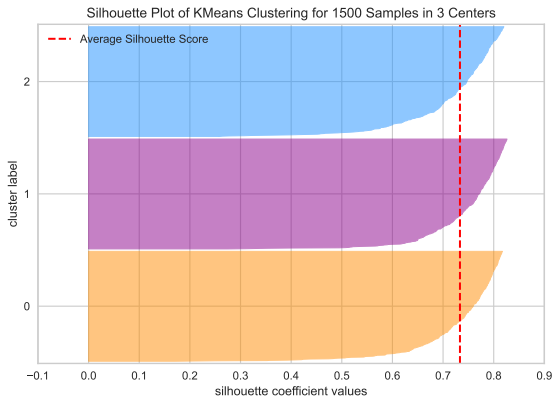
- ▶ higher means more structure
- ▶ $-1 \leq s(o) \leq 1$, where $s(o) = -1$ means a bad cluster assignment, 0 is indifferent and $+1$ a good assignment
- ▶ measure is independent of the number of clusters
- ▶ rule of thumb:
 - ▶ $s_c > 0.7$: strong structure,
 - ▶ $0.7 \geq s_c > 0.5$: usable structure,
 - ▶ $0.5 \geq s_c > 0.25$: weak structure,
 - ▶ $0.25 \geq s_c$: no structure.

 Admit when the coefficient is low,
that the algorithm found no real clustering.

Plotting the Silhouette

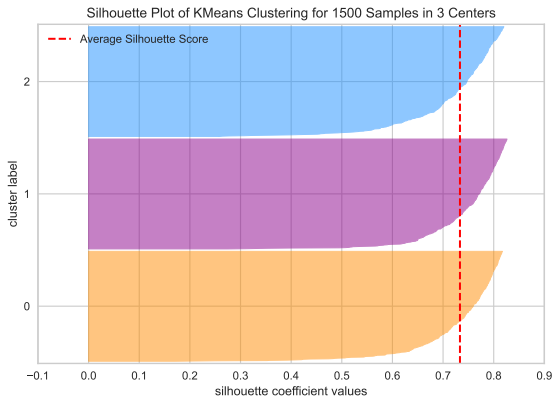


Plotting the Silhouette



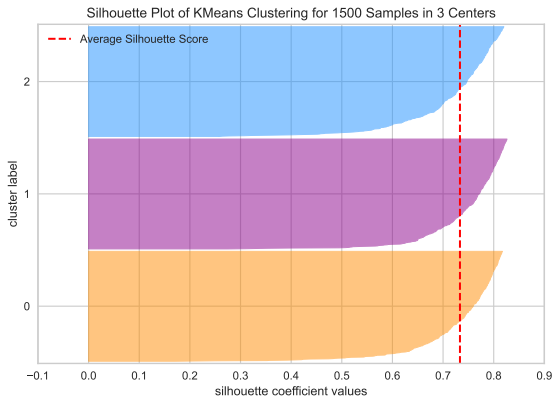
► silhouette score: $s = 0.73$

Plotting the Silhouette



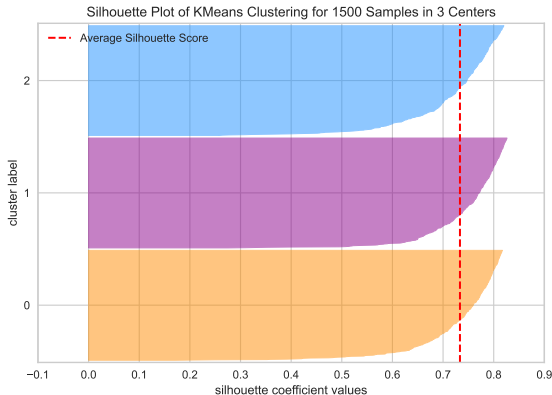
- silhouette score: $s = 0.73$
- one row for each data instance, ordered by cluster and silhouette

Plotting the Silhouette



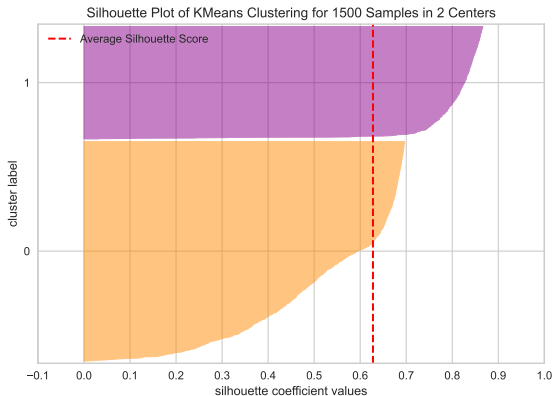
- silhouette score: $s = 0.73$
- one row for each data instance, ordered by cluster and silhouette
- we can assess quality per cluster

Plotting the Silhouette

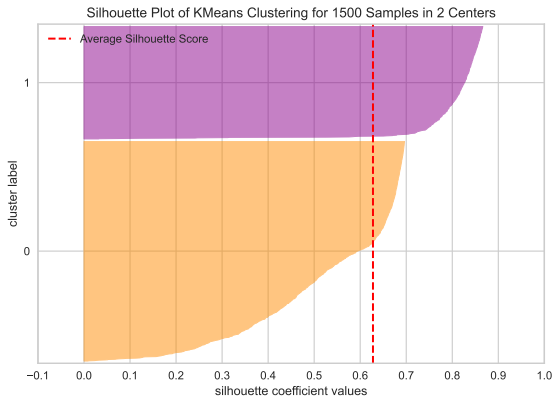


- ▶ silhouette score: $s = 0.73$
- ▶ one row for each data instance, ordered by cluster and silhouette
- ▶ we can assess quality per cluster
- ▶ visualization independent of dataset dimension!

Interpreting Silhouette Plots 1

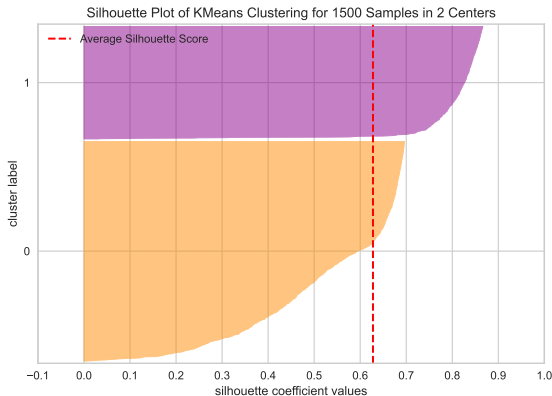


Interpreting Silhouette Plots 1



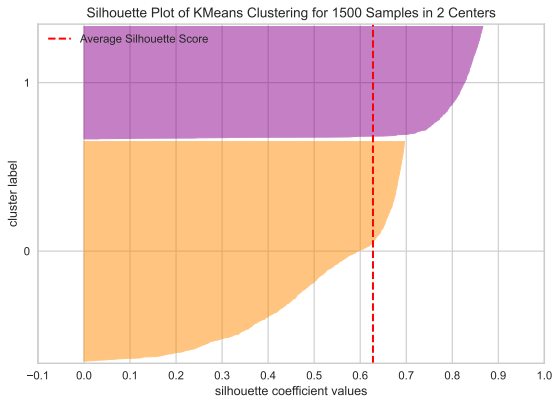
- ▶ overall structure not as strong as before ($s = 0.63$) (same dataset!)

Interpreting Silhouette Plots 1



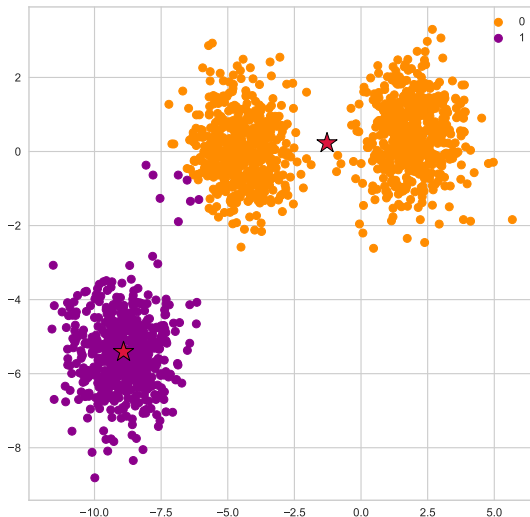
- ▶ overall structure not as strong as before ($s = 0.63$) (same dataset!)
- ▶ only two clusters

Interpreting Silhouette Plots 1

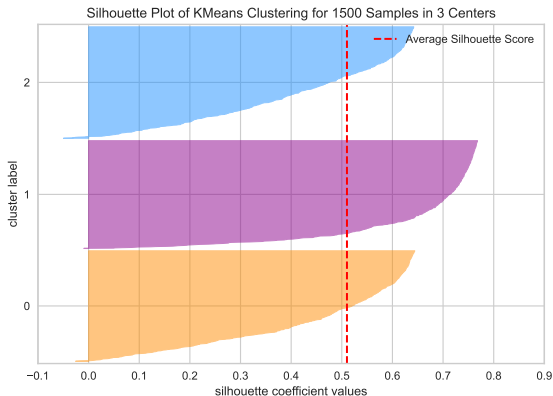


- ▶ overall structure not as strong as before ($s = 0.63$) (same dataset!)
- ▶ only two clusters
- ▶ cluster 0 seriously lacks structure (many instances with small silhouettes)

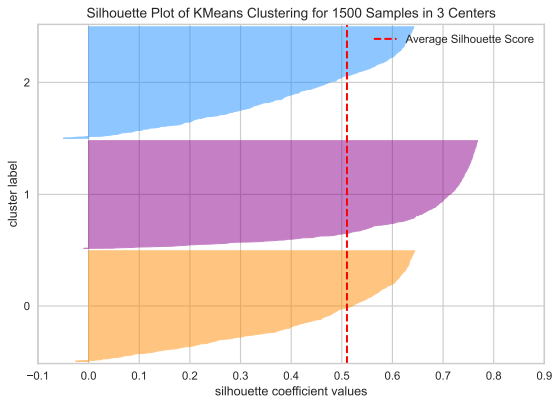
Interpreting Silhouette Plots 1



Interpreting Silhouette Plots 2

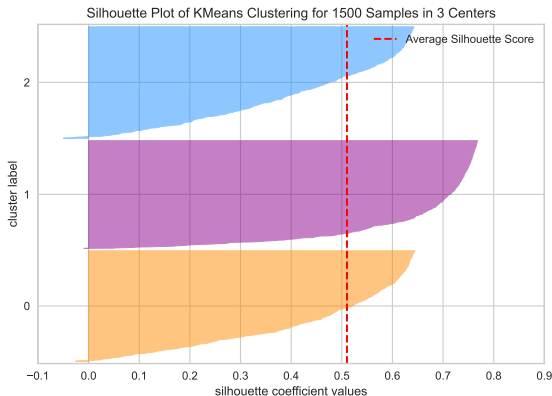


Interpreting Silhouette Plots 2



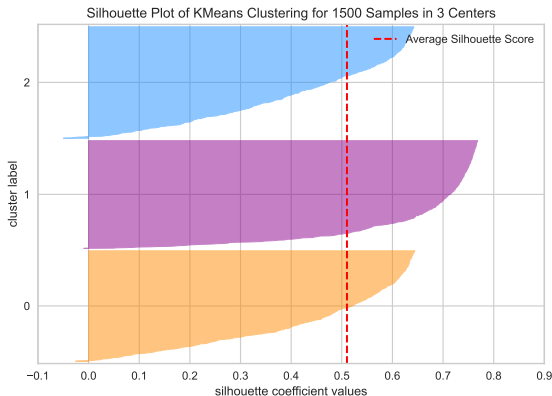
- overall structure close to unusable ($s = 0.51$)

Interpreting Silhouette Plots 2



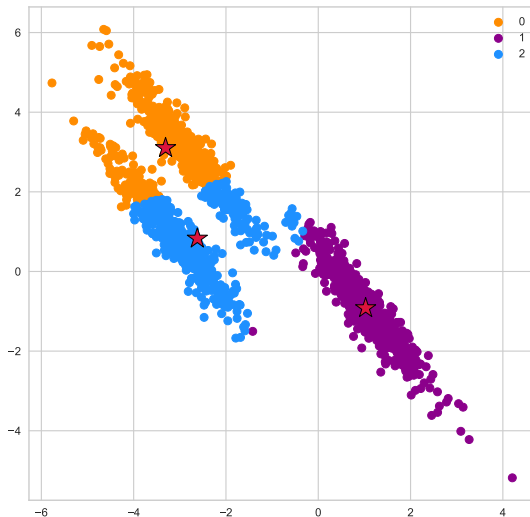
- ▶ overall structure close to unusable ($s = 0.51$)
- ▶ three cluster, one with good structure, two without

Interpreting Silhouette Plots 2

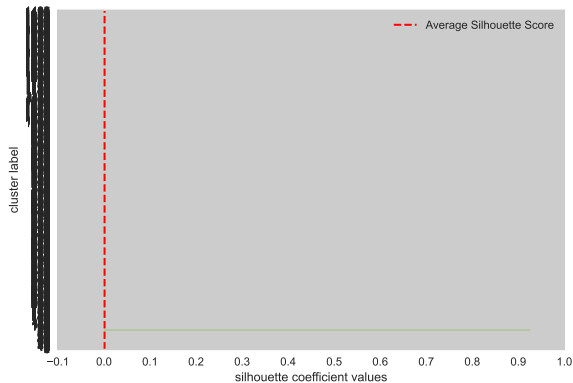


- ▶ overall structure close to unusable ($s = 0.51$)
- ▶ three cluster, one with good structure, two without
- ▶ some instances even have negative silhouette

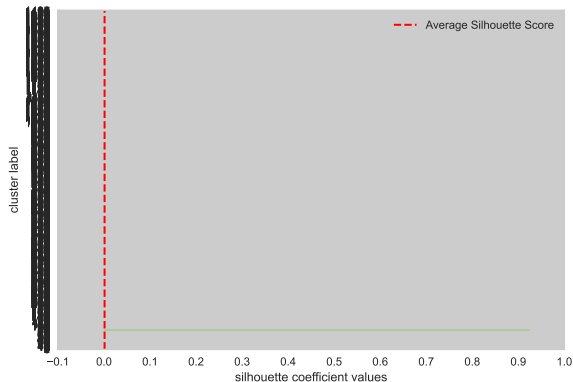
Interpreting Silhouette Plots 2



Interpreting Silhouette Plots 3

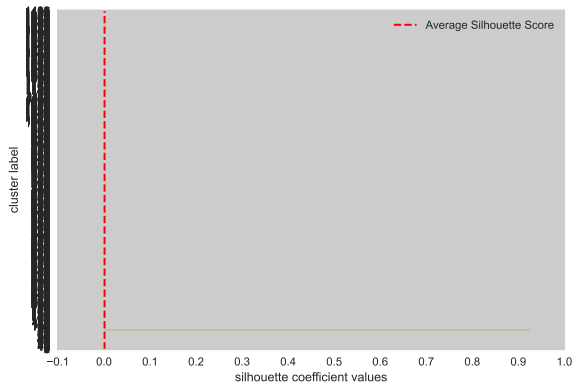


Interpreting Silhouette Plots 3



- silhouette coefficient is close to 0: $s = 0.0012$

Interpreting Silhouette Plots 3



- ▶ silhouette coefficient is close to 0: $s = 0.0012$
- ▶ 1500 instances and 1499 clusters

Interpreting Silhouette Plots 4

The silhouette plot:

- ▶ visualizes the data in 2D for arbitrary dimensional data!

Interpreting Silhouette Plots 4

The silhouette plot:

- ▶ visualizes the data in 2D for arbitrary dimensional data!
- ▶ shows the distribution of instances over the clusters (larger and smaller clusters).

Interpreting Silhouette Plots 4

The silhouette plot:

- ▶ visualizes the data in 2D for arbitrary dimensional data!
- ▶ shows the distribution of instances over the clusters (larger and smaller clusters).
- ▶ shows the overall silhouette coefficients.

Interpreting Silhouette Plots 4

The silhouette plot:

- ▶ visualizes the data in 2D for arbitrary dimensional data!
- ▶ shows the distribution of instances over the clusters (larger and smaller clusters).
- ▶ shows the overall silhouette coefficients.
- ▶ shows the distribution per cluster.

Interpreting Silhouette Plots 4

The silhouette plot:

- ▶ visualizes the data in 2D for arbitrary dimensional data!
- ▶ shows the distribution of instances over the clusters (larger and smaller clusters).
- ▶ shows the overall silhouette coefficients.
- ▶ shows the distribution per cluster.
- ▶ shows negative silhouettes, meaning points have closer average distance to points in another cluster than to their own.

Interpreting Silhouette Plots 4

The silhouette plot:

- ▶ visualizes the data in 2D for arbitrary dimensional data!
- ▶ shows the distribution of instances over the clusters (larger and smaller clusters).
- ▶ shows the overall silhouette coefficients.
- ▶ shows the distribution per cluster.
- ▶ shows negative silhouettes, meaning points have closer average distance to points in another cluster than to their own.
- ▶ can be used to discuss clusterings for different k on the same data.

Interpreting Silhouette Plots 4

The silhouette plot:

- ▶ visualizes the data in 2D for arbitrary dimensional data!
- ▶ shows the distribution of instances over the clusters (larger and smaller clusters).
- ▶ shows the overall silhouette coefficients.
- ▶ shows the distribution per cluster.
- ▶ shows negative silhouettes, meaning points have closer average distance to points in another cluster than to their own.
- ▶ can be used to discuss clusterings for different k on the same data.
- ▶ must be used with care when interpreting individual clusters!

Interpreting Silhouette Plots 4

The silhouette plot:

- ▶ visualizes the data in 2D for arbitrary dimensional data!
- ▶ shows the distribution of instances over the clusters (larger and smaller clusters).
- ▶ shows the overall silhouette coefficients.
- ▶ shows the distribution per cluster.
- ▶ shows negative silhouettes, meaning points have closer average distance to points in another cluster than to their own.
- ▶ can be used to discuss clusterings for different k on the same data.
- ▶ must be used with care when interpreting individual clusters!



When using k -means, ALWAYS discuss silhouette plots.
For other clustering algorithms, silhouettes often are
NOT meaningful.

Exercises

Exercises 1–2

Outline

Motivation

Basics

Construction of Central Points

- k -Means Algorithm

- Discussion of k -means

- Choosing a Good Cluster Number

Selecting Representative Instances

Idea

Assumption

data is grouped around a fix number of central instances, called
medoids

Idea

Assumption

data is grouped around a fix number of central instances, called **medoids**

Use Case

we want the center to be an actual representative (member) of the data, a prototypical element of the cluster

Idea

Assumption

data is grouped around a fix number of central instances, called
medoids

Use Case

we want the center to be an actual representative (member) of the data, a prototypical element of the cluster

Approach

Idea

Assumption

data is grouped around a fix number of central instances, called **medoids**

Use Case

we want the center to be an actual representative (member) of the data, a prototypical element of the cluster

Approach

- ▶ try determine those representatives

Idea

Assumption

data is grouped around a fix number of central instances, called **medoids**

Use Case

we want the center to be an actual representative (member) of the data, a prototypical element of the cluster

Approach

- ▶ try determine those representatives
- ▶ each instance is assigned to the representative it is closest to

Selecting Representative Instances

Instances: vectors $p = (x_{p_1}, \dots, x_{p_d})$ in a Euclidean vector space

Distance: Euclidean distance

Representative: **Medoid** $m_C \in D$ as central element of a cluster

Cluster-Cost: measure for the (non-)compactness of a Cluster C :

$$\text{cost}(C) = TD(C) := \sum_{p \in C} \text{dist}(p, m_C)$$

Clustering-Cost: measure for the (non-)compactness of a clustering:

$$\text{cost}(\mathcal{C}) = TD(\mathcal{C}) := \sum_{C \in \mathcal{C}} TD(C)$$

Search space of the clustering algorithm: all k -element partitions of D

→ runtime of exhaustive search: $O(|D|^k)$

Algorithms PAM and CLARANS

PAM

Algorithms PAM and CLARANS

PAM

- ▶ greedy-Algorithm: in each step exchange a medoid with a non-medoid

Algorithms PAM and CLARANS

PAM

- ▶ greedy-Algorithm: in each step exchange a medoid with a non-medoid
- ▶ choose the pair that reduces the cost TD the most

Algorithms PAM and CLARANS

PAM

- ▶ greedy-Algorithm: in each step exchange a medoid with a non-medoid
- ▶ choose the pair that reduces the cost TD the most

CLARANS two additional parameters: `maxneighbor` und `numlocal`

Algorithms PAM and CLARANS

PAM

- ▶ greedy-Algorithm: in each step exchange a medoid with a non-medoid
- ▶ choose the pair that reduces the cost TD the most

CLARANS two additional parameters: `maxneighbor` und `numlocal`

- ▶ at most `maxneighbor` many randomly chosen pairs (medoid, non-medoid) are considered

Algorithms PAM and CLARANS

PAM

- ▶ greedy-Algorithm: in each step exchange a medoid with a non-medoid
- ▶ choose the pair that reduces the cost TD the most

CLARANS two additional parameters: `maxneighbor` und `numlocal`

- ▶ at most `maxneighbor` many randomly chosen pairs (medoid, non-medoid) are considered
- ▶ the first replacement reducing TD is applied

Algorithms PAM and CLARANS

PAM

- ▶ greedy-Algorithm: in each step exchange a medoid with a non-medoid
- ▶ choose the pair that reduces the cost TD the most

CLARANS two additional parameters: `maxneighbor` und `numlocal`

- ▶ at most `maxneighbor` many randomly chosen pairs (medoid, non-medoid) are considered
- ▶ the first replacement reducing TD is applied
- ▶ search for k optimal medoids is repeated `numlocal` times

Algorithms PAM and CLARANS

PAM

- ▶ greedy-Algorithm: in each step exchange a medoid with a non-medoid
- ▶ choose the pair that reduces the cost TD the most

CLARANS two additional parameters: `maxneighbor` und `numlocal`

- ▶ at most `maxneighbor` many randomly chosen pairs (medoid, non-medoid) are considered
- ▶ the first replacement reducing TD is applied
- ▶ search for k optimal medoids is repeated `numlocal` times

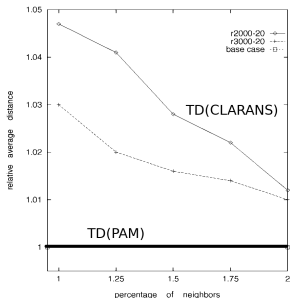
 Notebook 06_2_clarans_iris

Comparison of PAM and CLARANS

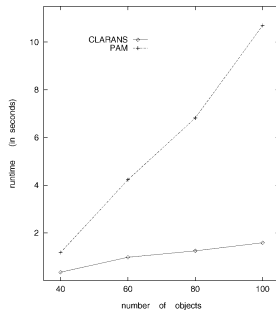
Runtime:

- ▶ PAM: $O(k(|D| - k)^2 \cdot \#iterations) (+|D|^3 \text{ for the "best" initialization})$
- ▶ CLARANS:
 $O(\text{numlocal} \cdot \text{maxneighbor} \cdot \#replacements \cdot |D|)$
: $O(|D|^2)$

Experimental Evaluation:



quality



runtime