MADS-ML – Machine Learning

Neighborhood-based Classification

Prof. Dr. Stephan Doerfel

FACHHOCHSCHULE KIEL
University of Applied Sciences

Moodle (WiSe 2024/25)

For a decision, remember similar instances and their decisions.

Outline

Basic Idea

The Neighborhood Size

The Decision Rule

Distance Functions

Python

Basic Idea 1/2

Assumption:

Basic Idea 1/2

Assumption:

Instances of the same class are **similar**. Similar instances are of the same class.

Basic Idea 1/2

Assumption:

Instances of the same class are **similar**. Similar instances are of the same class.

Idea:

Basic Idea 1/2

Assumption:

Instances of the same class are **similar**. Similar instances are of the same class.

Idea:

For a data instance x find those k **instances** from the training data that are the **most similar** to x.

Basic Idea 1/2

Assumption:

Instances of the same class are **similar**. Similar instances are of the same class.

Idea:

For a data instance x find those k **instances** from the training data that are the **most similar** to x.

Classification:

Basic Idea 1/2

Assumption:

Instances of the same class are **similar**. Similar instances are of the same class.

Idea:

For a data instance x find those k **instances** from the training data that are the **most similar** to x.

Classification:

Derive the class of x from the classes of these k instances.

Which ingredients do we need?

Basic Idea 2/2

"Lazy Learner":

- (as opposed to Eager Lerners)
- no explicit model is trained.
- the training data serves as the model
- no generalization from the training data

Basic Idea 2/2

"Lazy Learner":

- (as opposed to Eager Lerners)
- no explicit model is trained.
- the training data serves as the model
- no generalization from the training data

Distance-based:

distance function determines the (non-)similarity between to data instances

Basic Idea 2/2

"Lazy Learner":

- (as opposed to Eager Lerners)
- no explicit model is trained.
- the training data serves as the model
- no generalization from the training data

Distance-based:

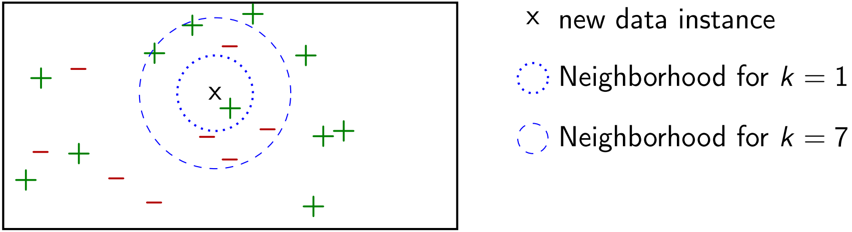
distance function determines the (non-)similarity between to data instances

How many neighbors:

select k nearest (most similar) neighbors

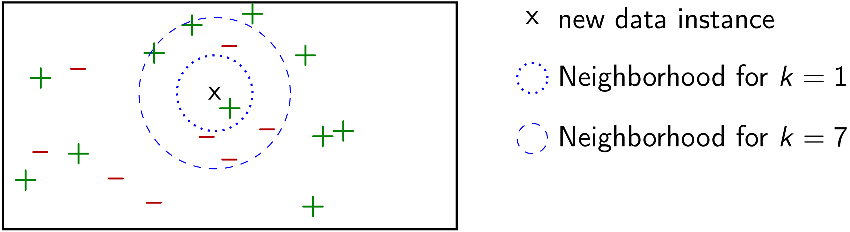
➔ Parameter $k \Rightarrow$ "k-nearest neighbor"-algorithm

kNN: Example



- training data: 11 instances of class "+" and 8 instances of class "–".
- $k = 1$: object x is classified
- $k = 7$ objekt x is classified

kNN: Example



- training data: 11 instances of class "+" and 8 instances of class "–".
- $k = 1$: object x is classified "+"
- $k = 7$ objekt x is classified "–"

kNN algorithm

Given a data instance x

- for each instance t of the training data
  - compute distance distance between x and t
- select k instances with the lowest distance
- apply the decision rule to those k instances

kNN Ingredients

Decision Data:

- the data set from which we choose the k nearest neighbors
- these are exactly the training data

kNN Ingredients

Decision Data:

- the data set from which we choose the k nearest neighbors
- these are exactly the training data

Neighborhood Size:

- hyperparameter k, the number of neighbors considered per instance

kNN Ingredients

Decision Data:

- the data set from which we choose the k nearest neighbors
- these are exactly the training data

Neighborhood Size:

- hyperparameter k, the number of neighbors considered per instance

Distance Function:

- defines the (non)-similariy for pairs of data instances
- many different choices depending on the data

kNN Ingredients

Decision Data:

- the data set from which we choose the k nearest neighbors
- these are exactly the training data

Neighborhood Size:

- hyperparameter k, the number of neighbors considered per instance

Distance Function:

- defines the (non)-similariy for pairs of data instances
- many different choices depending on the data

Decision Rule:

- How does one derive the class from the given k neighbors?

Outline

Basic Idea

The Neighborhood Size

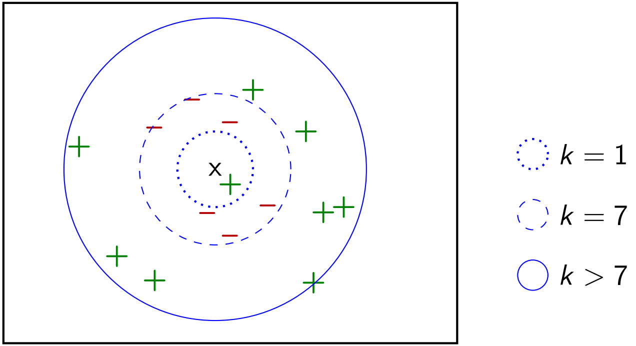
The Decision Rule

Distance Functions

Python

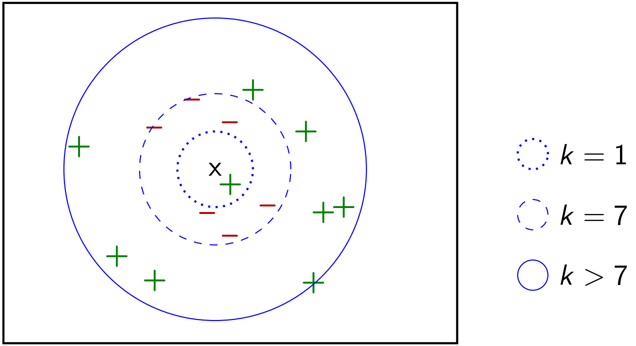
Choice of k

- k "too low":



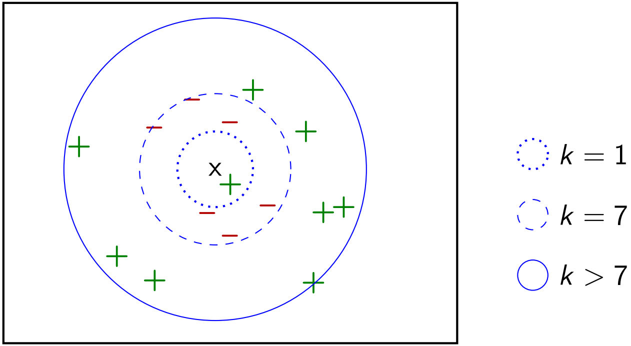
Choice of k

- k "too low": high sensitivity for outliers



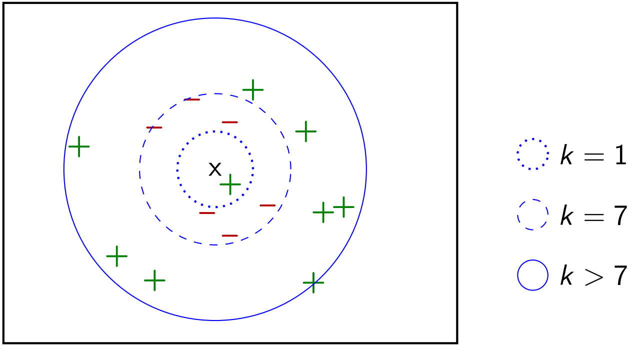
Choice of k

- k "too low": high sensitivity for outliers
- k "too high":



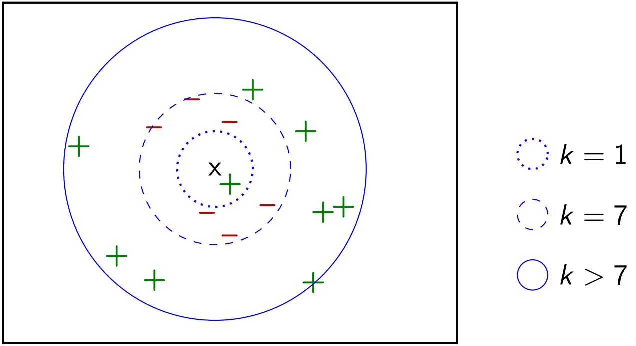
Choice of k

- k "too low": high sensitivity for outliers
- k "too high": many objects from other classes in the neighborhood



Choice of k

- k "too low": high sensitivity for outliers
- k "too high": many objects from other classes in the neighborhood
- rule of thumb: often $1 \ll k < 10$ yields highest classification quality



Simple determination of k

- Try different values $k = 1, 2, 3, \ldots$
- For each k: compute classification quality
  - ⌛ usually using cross validation
- use best k in production

Outline

Basic Idea

The Neighborhood Size

The Decision Rule

Distance Functions

Python

Decision rule

Decision rule

Standard:

- Use the majority class among the neighbors. Each neighbor has one vote with the weight 1.

Decision rule

Standard:

- Use the majority class among the neighbors. Each neighbor has one vote with the weight 1.

Weighted:

- Each neighbor votes with a weight:
  - by inverted distance
  - by inverted class frequency in the training dataset
  - by some cost function (e.g. when predicting + instead of actual – is more expansive than – instead of +.

Decision rule

Standard:

- Use the majority class among the neighbors. Each neighbor has one vote with the weight 1.

Weighted:

- Each neighbor votes with a weight:
  - by inverted distance
  - by inverted class frequency in the training dataset
  - by some cost function (e.g. when predicting + instead of actual – is more expansive than – instead of +.

Example weighted voting

Let the training set contain 95% of class A and only 5% of class B. A neighborhood of

$$\{A, A, A, A, B, B, B\}$$

yields: Standard $\Rightarrow A$, weighted by class $\Rightarrow B$.

kNN with probabilities

- probabilities help us judge "how sure" the classifier is in its classifications

kNN with probabilities

- probabilities help us judge "how sure" the classifier is in its classifications
- kNN offers probabilities based on the voting.

kNN with probabilities

- probabilities help us judge "how sure" the classifier is in its classifications
- kNN offers probabilities based on the voting.
- add the votes per class and divide by the number of neighbors

Example probabilities

Let the training set contain instances of classes A, B, and C. A neighborhood of

$$\{A, A, A, A, B, B, B\}$$

yields probabilities $p_A(x) = \frac{4}{7}, p_B(x) = \frac{3}{7},$ and $p_C(x) = 0$.

Outline

Basic Idea

The Neighborhood Size

The Decision Rule

Distance Functions

Python

Distance functions

Just a couple of examples. More on distances in MADS-MMS.

Definition 1 (Minkowski-Metrik)

Let $x = (x_1, \ldots, x_d)$ and $y = (y_1, \ldots, y_d)$ be numerical vectors. Then the **Minkowski-Metric** for p (L_p-metric) ist defined as:

$$\mathrm{dist}_p(x, y) := \sqrt[p]{\sum_{i=1}^{d} |x_i - y_i|^p}$$

Distance functions

Just a couple of examples. More on distances in MADS-MMS.

Definition 1 (Minkowski-Metrik)

Let $x = (x_1, \dots, x_d)$ and $y = (y_1, \dots, y_d)$ be numerical vectors. Then the **Minkowski-Metric** for p (L_p-metric) ist defined as:

$$\mathrm{dist}_p(x, y) := \sqrt[p]{\sum_{i=1}^{d} |x_i - y_i|^p}$$

Commonly used Minkowski-Metrics are:

- Euclidean Distance ($p = 2$): $\mathrm{dist}_2(x, y) = \sqrt[2]{\sum_{i=1}^{d} |x_i - y_i|^2}$
- Manhattan-Metric ($p = 1$): $\mathrm{dist}_1(x, y) = \sum_{i=1}^{d} |x_i - y_i|$
- Maximum-Metric ($p = \infty$):
  $\mathrm{dist}_\infty(x, y) = \max\{|x_i - y_i| \mid 1 \leq i \leq d\}$

A Problem with Distances

Imagine a dataset where the data instances are employees and the features are age, yearly salary and number of children. Using the Euclidean distance we yield:

$$\mathrm{dist}_2(x, y) = \sqrt[2]{(x_{age} - y_{age})^2 + (x_{kids} - y_{kids})^2 + (x_{salary} - y_{salary})^2}$$

A Problem with Distances

Imagine a dataset where the data instances are employees and the features are age, yearly salary and number of children. Using the Euclidean distance we yield:

$$\mathrm{dist}_2(x, y) = \sqrt[2]{(x_{age} - y_{age})^2 + (x_{kids} - y_{kids})^2 + (x_{salary} - y_{salary})^2}$$

- A difference of 1 is a lot for the number of kids, it is verly little for the yearly salary.

A Problem with Distances

Imagine a dataset where the data instances are employees and the features are age, yearly salary and number of children. Using the Euclidean distance we yield:

$$\mathrm{dist}_2(x, y) = \sqrt[2]{(x_{age} - y_{age})^2 + (x_{kids} - y_{kids})^2 + (x_{salary} - y_{salary})^2}$$

- A difference of 1 is a lot for the number of kids, it is verly little for the yearly salary.
- salary dominates the other features

A Problem with Distances

Imagine a dataset where the data instances are employees and the features are age, yearly salary and number of children. Using the Euclidean distance we yield:

$$\text{dist}_2(x, y) = \sqrt[2]{(x_{age} - y_{age})^2 + (x_{kids} - y_{kids})^2 + (x_{salary} - y_{salary})^2}$$

- A difference of 1 is a lot for the number of kids, it is verly little for the yearly salary.
- salary dominates the other features
- solution: **feature scaling** (normalization)

Feature Scaling (Min-Max)

Definition 2

Min-Max-Scaling a feature in a dataset D means computing

$$x' = \frac{x - x_{\text{min}}}{x_{\text{max}} - x_{\text{min}}}$$

where

- x is the original values of that feature for some data instance
- x' is the scaled value of x, and
- x_{min} and x_{max} are the lowest and highest values resp. in the dataset.

- Scaling maps the values of a feature into the interval

Feature Scaling (Min-Max)

Definition 2

Min-Max-Scaling a feature in a dataset D means computing

$$x' = \frac{x - x_{\text{min}}}{x_{\text{max}} - x_{\text{min}}}$$

where

- x is the original values of that feature for some data instance
- x' is the scaled value of x, and
- x_{min} and x_{max} are the lowest and highest values resp. in the dataset.

- Scaling maps the values of a feature into the interval $[0, 1]$

Feature Scaling (Min-Max)

Definition 2

Min-Max-Scaling a feature in a dataset D means computing

$$x' = \frac{x - x_{\text{min}}}{x_{\text{max}} - x_{\text{min}}}$$

where

- x is the original values of that feature for some data instance
- x' is the scaled value of x, and
- x_{min} and x_{max} are the lowest and highest values resp. in the dataset.

- Scaling maps the values of a feature into the interval $[0, 1]$
- Scaling depends on the dataset. Changes yield differently scaled instances.

Feature Scaling (Min-Max)

Definition 2

Min-Max-Scaling a feature in a dataset D means computing

$$x' = \frac{x - x_{\text{min}}}{x_{\text{max}} - x_{\text{min}}}$$

where

- x is the original values of that feature for some data instance
- x' is the scaled value of x, and
- x_{min} and x_{max} are the lowest and highest values resp. in the dataset.

- Scaling maps the values of a feature into the interval $[0, 1]$
- Scaling depends on the dataset. Changes yield differently scaled instances.
- Scaling is invertible.

Outline

Basic Idea

The Neighborhood Size

The Decision Rule

Distance Functions

Python

kNN in Python

sklearn.neighbors.KNeighborsClassifier

Parameters

- k
- distance: large variety of out of the box functions
- decision rules: standard and weighted by distance
  - ties are resolved depending on the order of the data
- parameters controlling memory and cpu consumption

Notebook 03_1_knn_digits

kNN in Python

sklearn.neighbors.KNeighborsClassifier

Parameters

- k
- distance: large variety of out of the box functions
- decision rules: standard and weighted by distance
  - ties are resolved depending on the order of the data
- parameters controlling memory and cpu consumption

Notebook 03_1_knn_digits

Exercises 1–3