Machine Learning

Lecture 2: k-Nearest Neighbors

Prof. Dr. Stephan Günnemann

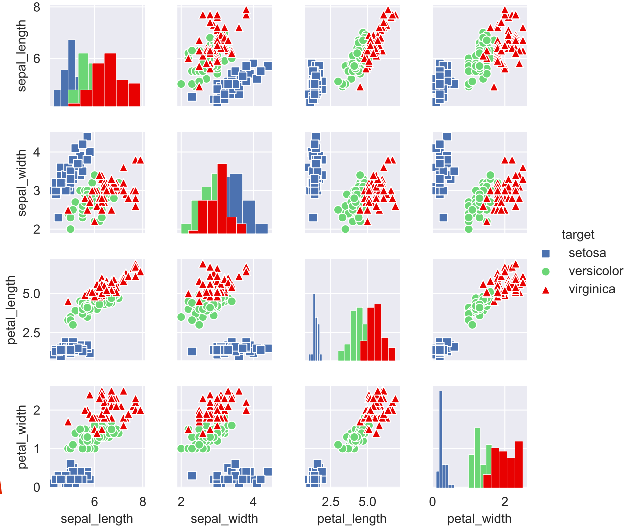
Data Analytics and Machine Learning
Technical University of Munich

Winter term 2022/2023

Iris dataset

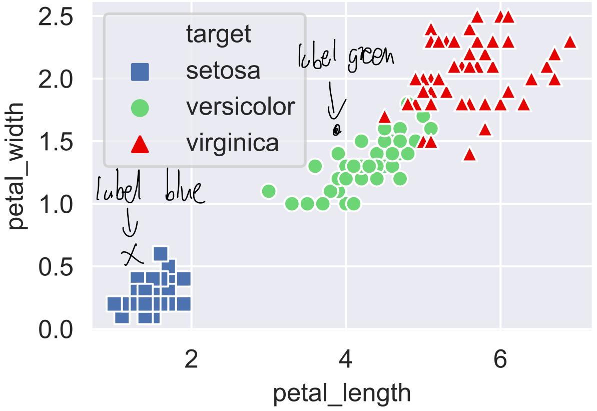
$f: x \to \{s, ve, vi\}$

视觉化数据-寻找规律



4-d

Iris dataset: 2 features



How do we intuitively label new samples by hand?
Look at the *surrounding* points. Do as your neighbor does.

1-NN algorithm

Given a training dataset $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{N}$
where $\boldsymbol{x}_i \in \mathbb{R}^D$ are features and $y_i \in \{1, \ldots, C\}$ are class labels

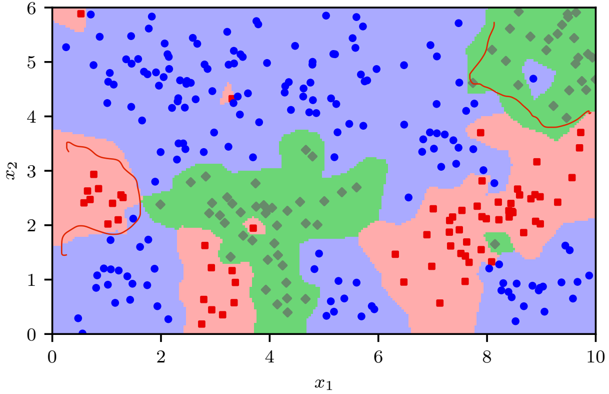
To classify new observations:

- define a distance measure (e.g. Euclidean distance)
- compute the nearest neighbor for all new data points
- and label them with the label of their nearest neighbor

This works for both *classification* and *regression*.

1-NN

not robus



This corresponds to a Voronoi tesselation.
And results in poor generalization...

k-Nearest Neighbor classification

look k clsoe neighbour

More *robust* against errors in the training set:

Look at multiple nearest neighbors and pick the majority label.

k-Nearest Neighbor classification

More *robust* against errors in the training set:

Look at multiple nearest neighbors and pick the majority label.

Let $\mathcal{N}_k(\boldsymbol{x})$ be the k nearest neighbors of a vector $\boldsymbol{x}$, then in classification tasks:

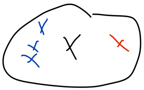
$$p(\underset{\uparrow\ \text{prob of belonging in class c}}{y} = c \mid \overset{\text{input vector}}{\underset{}{\overset{\downarrow}{\boldsymbol{x}}}}, \overset{\text{NN}}{\overset{\downarrow}{k}}) = \frac{1}{k} \sum_{i \in \mathcal{N}_k(\boldsymbol{x})} \mathbb{I}(y_i = c),$$

$\Leftarrow$ $\mathcal{N}_k(\boldsymbol{x})$ be the k nearest neighbors of a vector $\boldsymbol{x}$

$$\hat{y} = \underline{\arg\max_c p}(y = c \mid \boldsymbol{x}, k)$$

with the *indicator variable* $\mathbb{I}(e)$ is defined as:

with the *indicator variable* $\mathbb{I}(e) = \begin{cases} 1 \text{ if } e \text{ is true} \\ 0 \text{ if } e \text{ is false.} \end{cases}$



$p(y=B) = \frac{3}{4}$

$p(y=R) = \frac{1}{4}$

k-Nearest Neighbor classification

More *robust* against errors in the training set:

Look at multiple nearest neighbors and pick the majority label.

Let $\mathcal{N}_k(\boldsymbol{x})$ be the k nearest neighbors of a vector $\boldsymbol{x}$, then in classification tasks:

$$p(y = c \mid \boldsymbol{x}, k) = \frac{1}{k} \sum_{i \in \mathcal{N}_k(\boldsymbol{x})} \mathbb{I}(y_i = c),$$

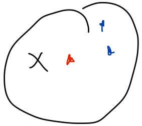
$$\hat{y} = \arg\max_c p(y = c \mid \boldsymbol{x}, k)$$

with the *indicator variable* $\mathbb{I}(e)$ is defined as:

$$\mathbb{I}(e) = \begin{cases} 1 \text{ if } e \text{ is true} \\ 0 \text{ if } e \text{ is false.} \end{cases}$$

i.e., the vector will be labeled by the mode of its neighbors' labels.

k-Nearest Neighbor classification: weighted



Look at multiple nearest neighbors and pick the weighted majority label.

可能是蓝色的，因为蓝色多。也可能是红色的，因为离红色近

k-Nearest Neighbor classification: weighted

Look at multiple nearest neighbors and pick the weighted majority label.
The weight is inversely proportional to the distance.

Let $\mathcal{N}_k(\boldsymbol{x})$ be the k nearest neighbors of a vector $\boldsymbol{x}$, then in classification tasks:

distance measure

$$p(y = c \mid \boldsymbol{x}, k) = \frac{1}{Z} \sum_{i \in \mathcal{N}_k(\boldsymbol{x})} \boxed{\frac{1}{\mathrm{d}(\boldsymbol{x}, \boldsymbol{x}_i)}} \mathbb{I}(y_i = c),$$

$d \downarrow$ weight $\uparrow$

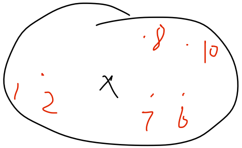
$$\hat{y} = \arg\max_c p(y = c \mid \boldsymbol{x}, k)$$

with $Z = \sum_{i \in \mathcal{N}_k(x)} \frac{1}{\mathrm{d}(\boldsymbol{x}, \boldsymbol{x}_i)}$ the normalization constant and $\mathrm{d}(\boldsymbol{x}, \boldsymbol{x}_i)$ being a distance measure between $\boldsymbol{x}$ and $\boldsymbol{x}_i$.

$$\underline{\text{with } Z = \sum_{i \in \mathcal{N}_k(x)} \frac{1}{\mathrm{d}(\boldsymbol{x}, \boldsymbol{x}_i)}}$$

Normalization constant
$\Rightarrow$ Sum = 1

k-Nearest-Neighbor regression



Regression is similar:

Let $\mathcal{N}_k(\boldsymbol{x})$ be the k nearest neighbors of a vector $\boldsymbol{x}$, then for regression:

$$\hat{y} = \frac{1}{Z} \sum_{i \in \mathcal{N}_k(\boldsymbol{x})} \frac{1}{\mathrm{d}(\boldsymbol{x}, \boldsymbol{x}_i)} \boxed{y_i},$$

with $Z = \sum_{i \in \mathcal{N}_k(x)} \frac{1}{\mathrm{d}(\boldsymbol{x}, \boldsymbol{x}_i)}$ the normalization constant and $\mathrm{d}(\boldsymbol{x}, \boldsymbol{x}_i)$ being a distance measure between $\boldsymbol{x}$ and $\boldsymbol{x}_i$,

k-Nearest-Neighbor regression

Regression is similar:

Let $\mathcal{N}_k(\boldsymbol{x})$ be the k nearest neighbors of a vector $\boldsymbol{x}$, then for regression:

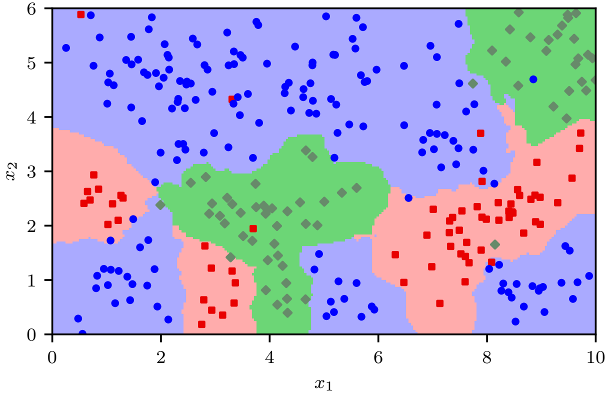
$$\hat{y} = \frac{1}{Z} \sum_{i \in \mathcal{N}_k(\boldsymbol{x})} \frac{1}{\mathrm{d}(\boldsymbol{x}, \boldsymbol{x}_i)} \, y_i,$$

with $Z = \sum_{i \in \mathcal{N}_k(x)} \frac{1}{\mathrm{d}(\boldsymbol{x}, \boldsymbol{x}_i)}$ the normalization constant and $\mathrm{d}(\boldsymbol{x}, \boldsymbol{x}_i)$ being a distance measure between $\boldsymbol{x}$ and $\boldsymbol{x}_i$,

i.e., the vector will be labeled by a weighted mean of its neighbors' values.

Note: y_i is a real number here (rather then categorical label).

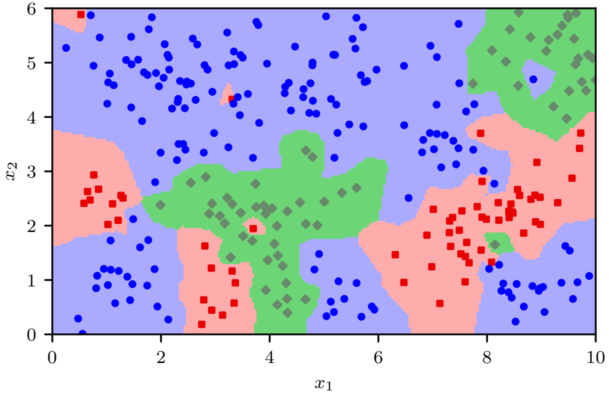
3-NN



So, how many neighbors are best?

1-NN

if k equal = training set, all region will be majority

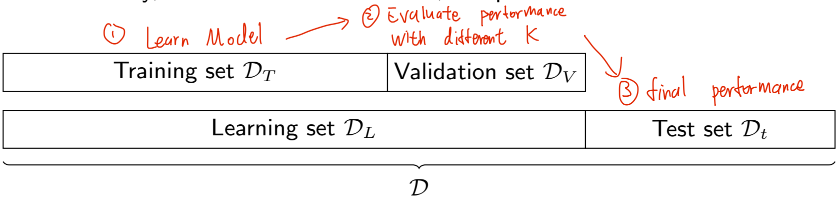


Compare the decision boundaries of 1-NN and 3-NN

Choosing k

Goal is generalization: pick k (called a *hyper-parameter*) that performs best[1] on unseen (future) data.

Unfortunately, no access to future data, so split the dataset $\mathcal{D}$:



① Learn Model
② Evaluate performance with different K
③ final performance
Training set $\mathcal{D}_T$
Validation set $\mathcal{D}_V$
Learning set $\mathcal{D}_L$
Test set $\mathcal{D}_t$
$\mathcal{D}$

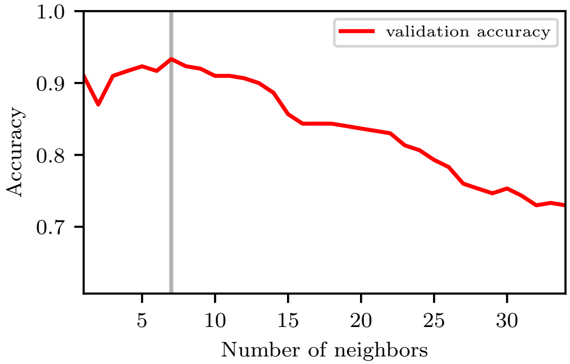
Hyper-parameter tuning procedure

- Learn the model using the training set stratified sampling
- Evaluate performance with different k on the *validation set* picking the best k
- Report final performance on the test set.[2]

如果一个验证集没有蓝色，这是非常不好的。是用下面的技巧，可以保证80:10:10

[1] In terms of some predefined metric, e.g., accuracy
[2] Good data science practices: See slides on Decision Trees

Using validation set to choose k



We choose $k = 7$.

Measuring classification performance

How can we assess the performance of a (binary) classification algorithm?

$\Rightarrow$ Confusion table

Predicted	True condition $y = 1$	$y = 0$
$y = 1$	TP	FP
$y = 0$	FN	TN

$\checkmark$.

TP = true positive
TN = true negative } correct predictions

FP = false positive
FN = false negative } wrong predictions

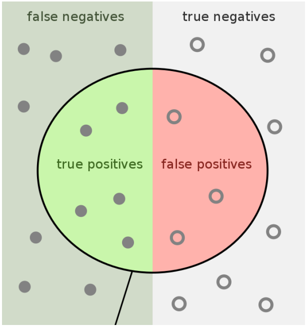
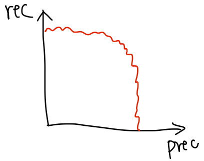


Image source: `https://en.wikipedia.org/wiki/Precision_and_recall`

Measuring classification performance

Accuracy:	$\mathrm{acc} = \frac{TP + TN}{TP + TN + FP + FN} = \frac{\text{correct}}{\text{ALL}}$
Precision:	$\mathrm{prec} = \frac{TP}{TP + FP}$
Sensitivty/Recall:	$\mathrm{rec} = \frac{TP}{TP + FN}$
Specificity:	$\mathrm{tnr} = \frac{TN}{FP + TN}$
False Negative Rate:	$\mathrm{fnr} = \frac{FN}{TP + FN}$
False Positive Rate:	$\mathrm{fpr} = \frac{FP}{FP + TN}$
F1 Score:	$f1 = \frac{2 \cdot \mathrm{prec} \cdot \mathrm{rec}}{\mathrm{prec} + \mathrm{rec}}$



$\Rightarrow$ Trade-off between precision and recall: increasing one (most often) leads to decreasing the other

需要设置权重

General note: Be careful when you have imbalanced classes! 10 9990

Distance measures

- K-NN can be used with various distance measures $\rightarrow$ highly flexible
- Euclidean distance (L_2 norm): $\sqrt{\sum_i (u_i - v_i)^2}$

$x \in \mathbb{R}^2$

Distance measures

- K-NN can be used with various distance measures $\rightarrow$ highly flexible
- Euclidean distance (L_2 norm): $\sqrt{\sum_i (u_i - v_i)^2}$
- L_1 norm: $\sum_i |u_i - v_i|$

Distance measures

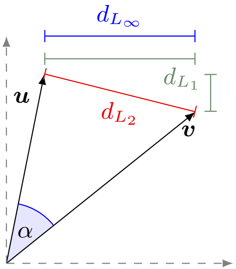
- K-NN can be used with various distance measures $\rightarrow$ highly flexible
- Euclidean distance (L_2 norm): $\sqrt{\sum_i (u_i - v_i)^2}$

- L_1 norm: $\sum_i |u_i - v_i|$
- L_∞ norm: $\max_i |u_i - v_i|$

Distance measures

- K-NN can be used with various distance measures $\rightarrow$ highly flexible
- Euclidean distance (L_2 norm): $\sqrt{\sum_i (u_i - v_i)^2}$
- L_1 norm: $\sum_i |u_i - v_i|$
- L_∞ norm: $\max_i |u_i - v_i|$
- Angle:

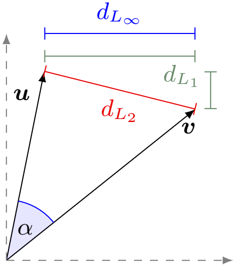
$$\cos\alpha = \frac{\boldsymbol{u}^T \boldsymbol{v}}{\|\boldsymbol{u}\| \|\boldsymbol{v}\|}$$



Distance measures

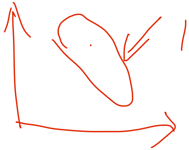
- K-NN can be used with various distance measures $\rightarrow$ highly flexible
- Euclidean distance (L_2 norm): $\sqrt{\sum_i (u_i - v_i)^2}$
- L_1 norm: $\sum_i |u_i - v_i|$
- L_∞ norm: $\max_i |u_i - v_i|$
- Angle:

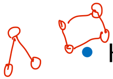
$$\cos\alpha = \frac{\boldsymbol{u}^T \boldsymbol{v}}{\|\boldsymbol{u}\| \|\boldsymbol{v}\|}$$



- *Mahalanobis distance* ($\boldsymbol{\Sigma}$ is *positive (semi) definite* and *symmetric*):

$$\sqrt{(\boldsymbol{u} - \boldsymbol{v})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{u} - \boldsymbol{v})}$$

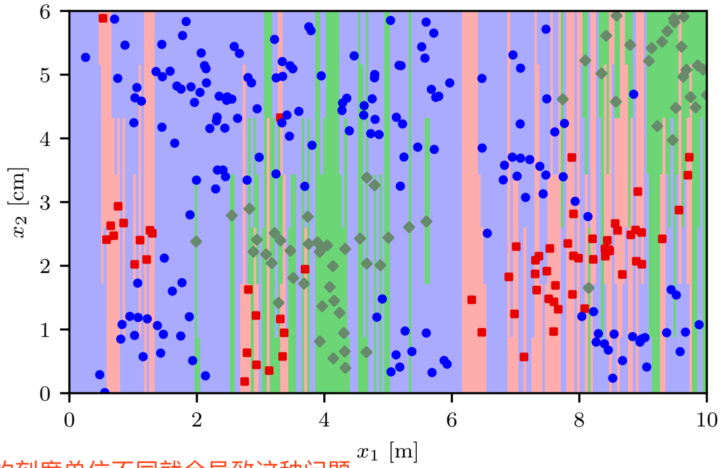




- Hamming distance, Edit distance, ...

used for txt

Scaling issues



两轴的刻度单位不同就会导致这种问题

The same old example but one of our features is in the order of meters, the other in the order of centimeters. $(k = 1)$

Circumventing scaling issues

- Data *standardization*
  Scale each feature to zero mean and unit variance.

  $$x_{i,\text{std}} = \frac{x_i - \mu_i}{\sigma_i}$$

  (This is a standard procedure in machine learning. Many models are sensitive to differences in scale.)

Circumventing scaling issues

min-max Nomilization. $\sim [0, 1]$

- Data *standardization*
  Scale each feature to zero mean and unit variance.

$$x_{i,\text{std}} = \frac{x_i - \mu_i}{\sigma_i}$$

Z - score

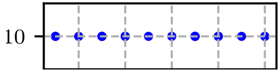
  (This is a standard procedure in machine learning. Many models are sensitive to differences in scale.)

- Use the Mahalanobis distance.

$$\text{mahalanobis}(\boldsymbol{x}_1, \boldsymbol{x}_2) = \sqrt{(\boldsymbol{x}_1 - \boldsymbol{x}_2)^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{x}_1 - \boldsymbol{x}_2)}$$

$$\boxed{\boldsymbol{\Sigma} = \begin{bmatrix} \sigma_1^2 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \sigma_n^2 \end{bmatrix}}$$ is equal to Euclidean distance on normalized data

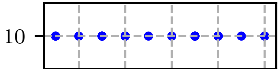
The curse of dimensionality

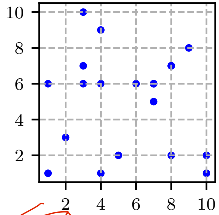


Given a discrete one-dimensional input space $x \in \{1, 2, \ldots, 10\}$

For $N = 20$ uniformly distributed samples the data covers 100% of the input space.

The curse of dimensionality



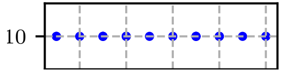


Given a discrete one-dimensional input space $x \in \{1, 2, \ldots, 10\}$

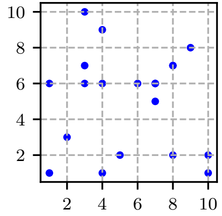
For $N = 20$ uniformly distributed samples the data covers 100% of the input space.

Add a second dimension (now $\boldsymbol{x} \in \{1, \ldots, 10\}^2$) and your data only covers 18% of the input space.

The curse of dimensionality

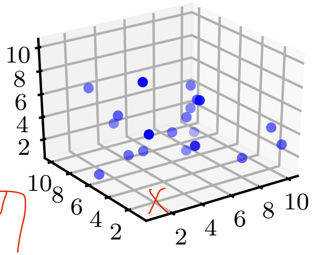


Given a discrete one-dimensional input space $x \in \{1, 2, \ldots, 10\}$



For $N = 20$ uniformly distributed samples the data covers 100% of the input space.

Add a second dimension (now $\boldsymbol{x} \in \{1, \ldots, 10\}^2$) and your data only covers 18% of the input space.



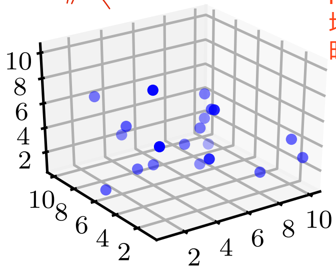
Once you add a third dimension you only cover 2%.

The curse of dimensionality

collect more data

- The nearest neighbor will now be pretty far away..
- N has to grow exponentially with the number of features. Consider this when using k-NN on high-dimensional data.

N要随着特征的数量呈指数级增长。在高维数据上使用k-NN时要考虑到这一点。



Practical considerations

limitation

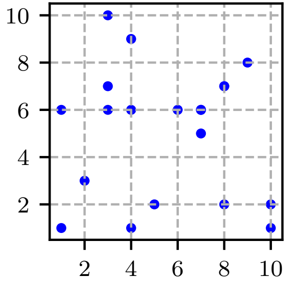
Expensive: memory and naive inference are both $\boxed{O(N)}$:

we need to store the entire training data and compare with all training instances to find the nearest neighbor

Solution: use tree-based search structures (e.g. k-d tree) for efficient (approximate) NN [3]

lazy when get new x

→ classifaction.



[3]At the expense of an additional computation performed only once

What we learned

- k-NN Algorithm
- Train-validation-test split
- Measuring classification performance
- Distance metrics
- Curse of dimensionality

Reading material

Main reading

- "Machine Learning: A Probabilistic Perspective" by Murphy
  [ch. 1.4.1 - 1.4.3]

Extra reading

- "Bayesian Reasoning and Machine Learning" by Barber
  [ch. 14]

Slides adapted from previous versions by W. Koepp & D. Korhammer