

## 2.2.3) Der KNN - Algorithmus

Bisher: Klassifikation mit Hilfe probabilistischer Modelle

- Logistische Regression
- Lineare Diskriminanzanalyse

Zieht: einfacher, denkbare ML-Algorithmus → KNN

**FRAGE** Besten Classifier, der möglich ist?

### **BAYES - CLASSIFIER**

Ordne einer Beobachtung  $(x_0, y_0)$  aus der Testmenge mit Feature-Vektor  $x_0$  diejenige Klasse  $j$  zu, für die gilt:

$P(Y=j | X=x_0)$  ist maximal

bedingte Wahrscheinlichkeit  
 $\leftarrow Y=j$  zu beobachten unter der  
 Bedingung  $X=x_0$

⇒ Hat minimalen Fehler!

PROBLEM: Die zugrundeliegende Verteilung ist unbekannt! 153

Wir können nur versuchen, sie zu approximieren ...

Dazu könnten wir Verteilungsannahmen machen, aber das ist hier gar nicht nötig

⇒ kNN ist ein nicht-parametrisches Verfahren zum Schätzen der zugrundeliegenden W-Verteilung.

AUSGANGSLAGE:  $(x_i, y_i)$  ( $i=1\dots,n$ ) mit Feature-Vektor  $x_i$  und Target Y (m-Klassen)



Ordne einen Datenpunkt derjenigen Klasse zu, zu der die meisten Punkte in seiner Nachbarschaft gehören.

→ Dazu brauchen wir  
- Def. von "Nachbarschaft"  
- Metrik für die Abstands-  
messung

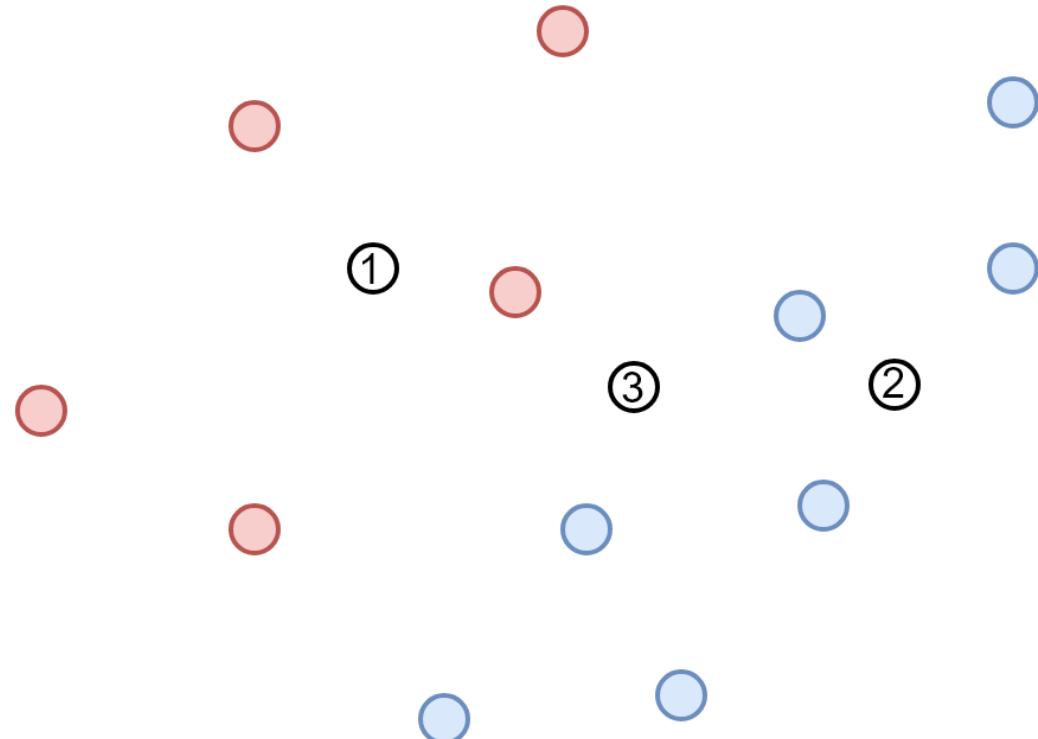
Bsp.: Zwei Features  $x_1, x_2$   
Zwei Klassen ◉ ●



① gehört zu Klasse ◉

② gehört zu Klasse ●

③ unentschlossen 😐



Beobachtung: Um konkret zu werden brauchen wir eine sinnvolle Metrik!

$x, x' \in \mathbb{R}^d$

$$\text{dist}(x, x') = \|x - x'\|_2 = \sqrt{\sum_{i=1}^d (x_i - x'_i)^2}$$

← EUKLIDISCHER ABSTAND  
(kanonisch)

damit

Def.: Nachbarschaft

Sei  $x \in \mathbb{R}^d$ . Dann ist eine ( $k$ -elementige) Nachbarschaft von  $x$

definiert als:

$$W^k(x) = \{p_1, \dots, p_k\}$$

"diejenigen  $k$  Punkte in  $D$ , die den kleinsten Abstand zu  $x$  haben"

$W^k(x)$  hat offensichtlich die Mächtigkeit  $|W^k(x)| = k$ .

Mit Hilfe der Nachbarschaft  $W^k(x)$  können wir eine simple Schätzung der bedingten Wahrscheinlichkeit

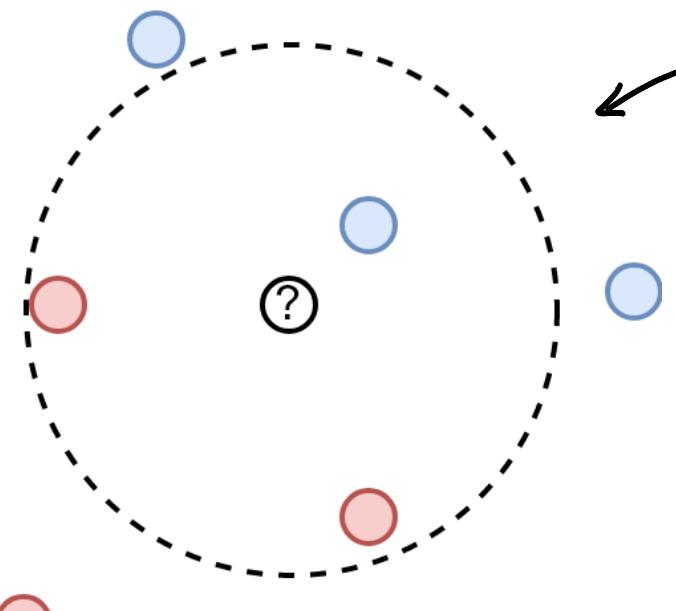
$$P(Y=j | X=x)$$

vornehmen, nämlich

in Datenmenge  
 $D$

$$P(Y=j | X=x) \approx \frac{\text{"Anzahl der } p_i \in W^k(x) \text{ mit Klasse } j"}{|W^k(x)|}$$

156



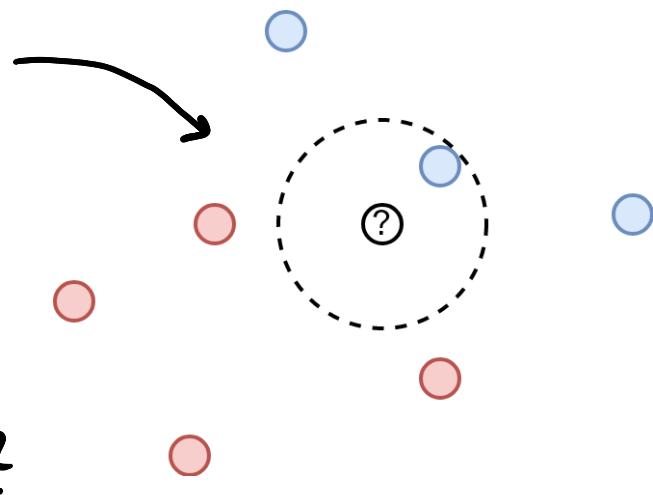
$$P(Y=\text{rot} | X=x) = \frac{2}{3}$$

$$P(Y=\text{blau} | X=x) = \frac{1}{3}$$

$\Rightarrow$  Die Wahrscheinlichkeit für  $Y=\text{rot}$  ist größer, also wird  $\textcircled{3}$  wohl eher rot sein...

$$P(Y=\text{rot} | X=x) = 0$$

$$P(Y=\text{blau} | X=x) = 1$$



stimmig ??

Logisch: Die Größe der Nachbarschaft ist wichtig!

Was passiert bei mehreren Klassen ??

15+

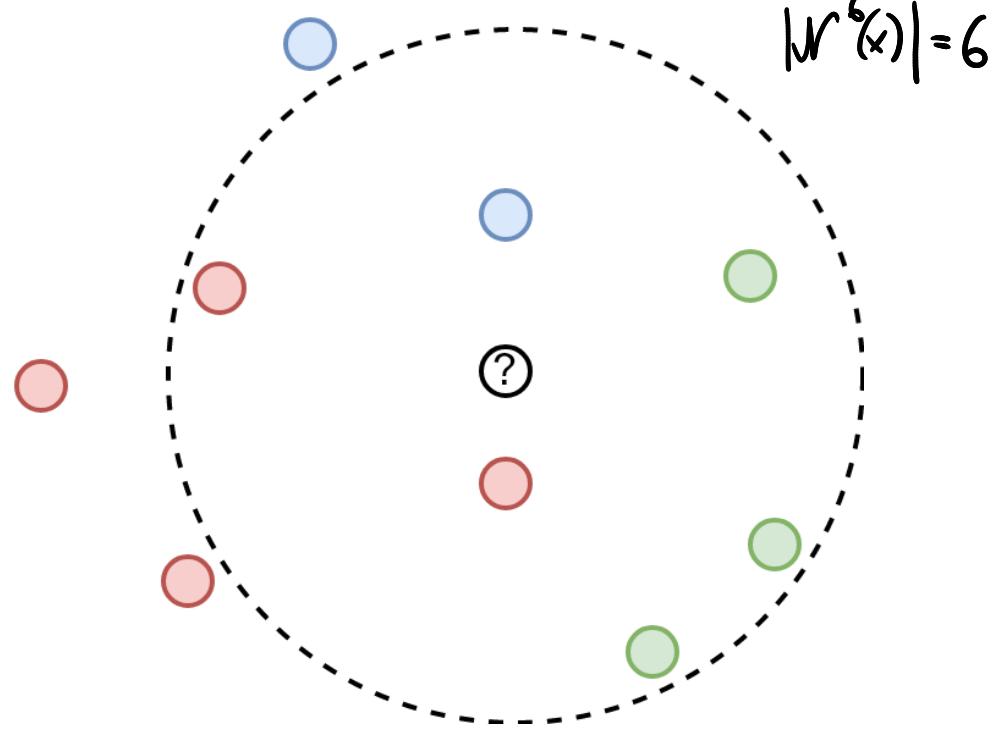
→ Lösung analog:

$$P(Y=\text{rot} | X=x) = \frac{2}{6} = \frac{1}{3}$$

$$P(Y=\text{blau} | X=x) = \frac{1}{6}$$

$$P(Y=\text{grün} | X=x) = \frac{3}{6}$$

⇒ ④ wahrscheinlich ●



Die Schätzung der bedingten Wahrscheinlichkeit  
erfolgt ohne eine Annahme an die darunterliegende W-Verteilung. ← kein Parameter zu schätzen

Es gibt aber einen Meta-Parameter im Modell

$$\Rightarrow |N^k(x)| = k \quad \leftarrow \text{Größe der verwendeten Nachbarschaft}$$

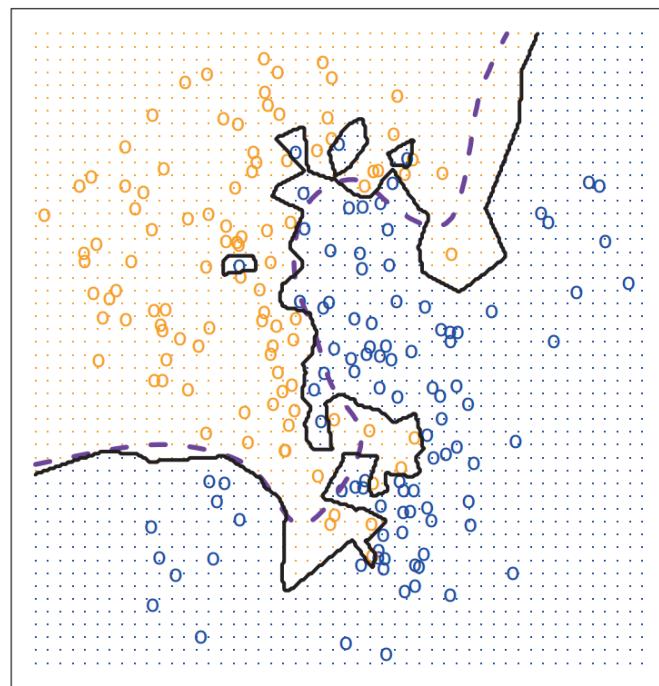
Schönes Beispiel:

Daten mit bekannte Verteilung  
synthetisiert ...

[Tibshirani et.al. (2014) , An Introduction to Statistical Learning ]

----- ideal  
——— Modell

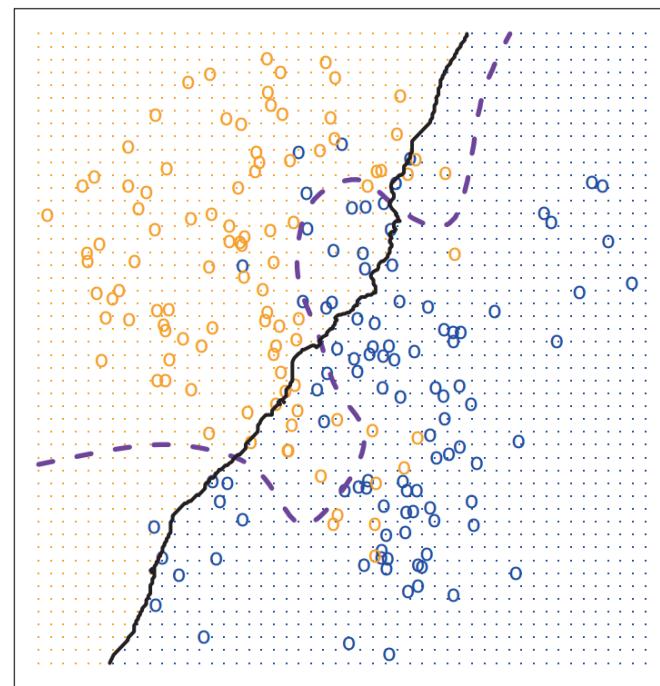
KNN: K=1



$k=1$  zu klein!

$\Rightarrow$  keine Generalisierungsfähigkeit!  
große Varianz  $\Rightarrow$  OVERTFITTING

KNN: K=100



----- ideal  
——— Modell

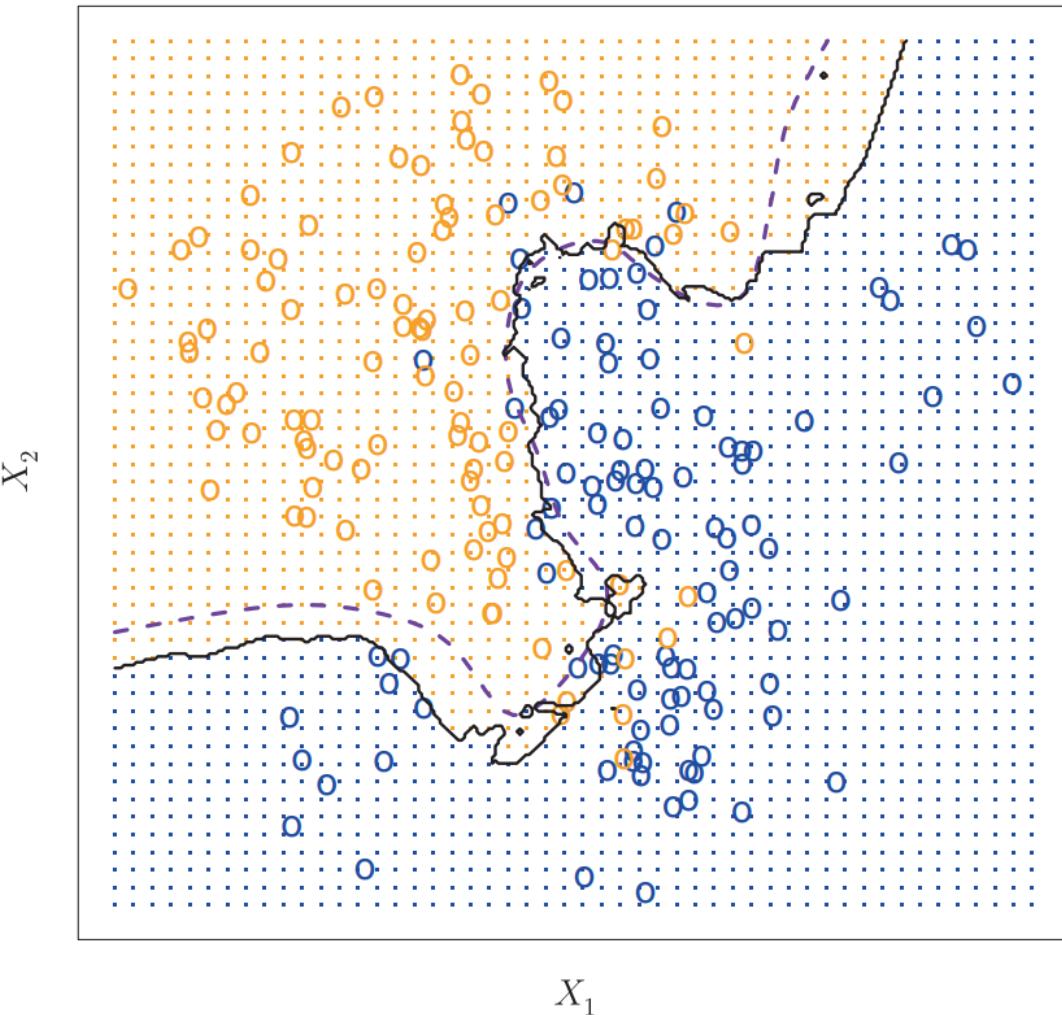
$\Rightarrow$  zu geringe Modellkomplexität

großer Bias  $\Rightarrow$  UNDERFITTING

$k=100$   
zu groß!

KNN: K=10

---- ideal  
— Modell



$k=10$  passt ganz gut!

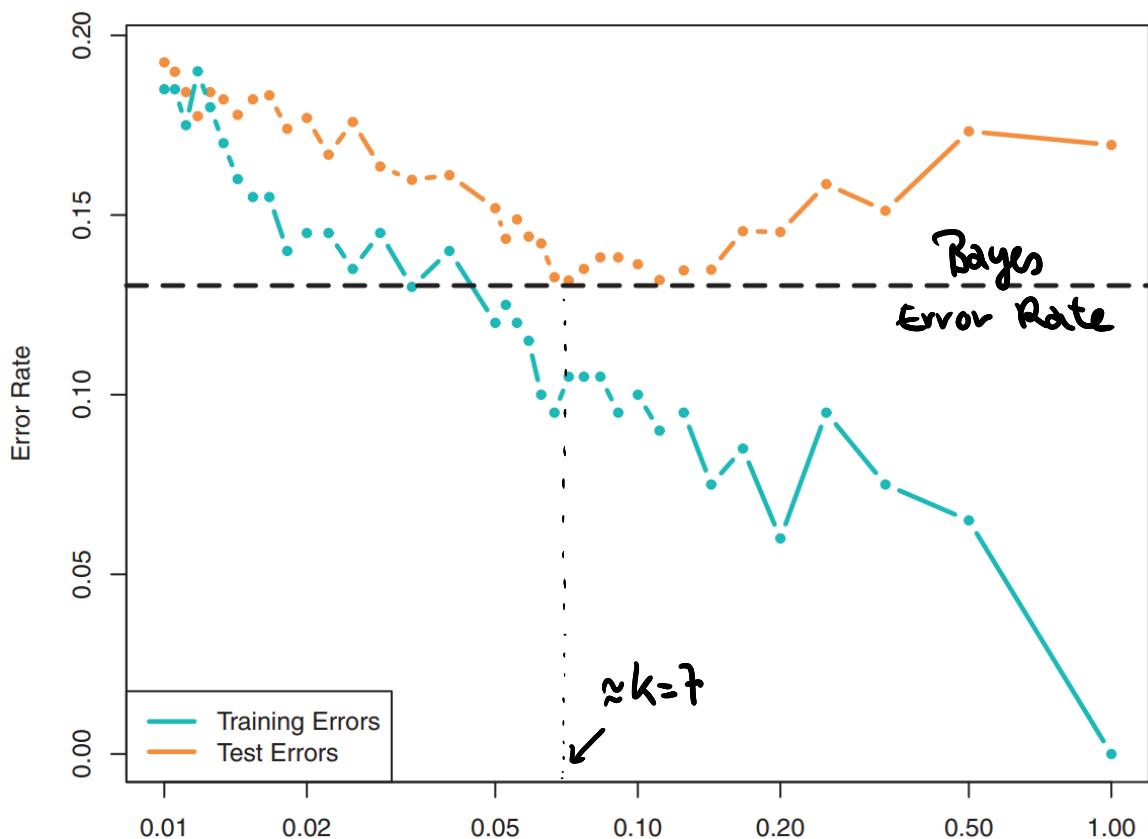
→ die gelernte Entscheidungsgrenze approximiert die optimale Lösung (Bayes)  
sehr gut  
(Optimum im Sinne des Bias-Variance Dilemmas!)



Wie finden wir das optimale  $k \in \mathbb{N}$ ?

→ Variiere  $k$  und werte die Error Rate =  $\frac{\text{"Anzahl Fehlklassifikationen"}}{\text{"Anzahl aller Datenpunkte"}}$

auf der Trainings- und Testmenge aus ( $\rightarrow$  Visualisierung, Bias-Variance Dilemma)



$$\overset{\uparrow}{k=100} \Rightarrow |U^t(x)| = 100$$

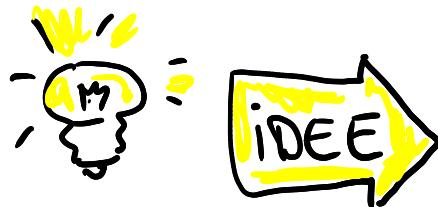
$$|U^t(x)| = 1 \Rightarrow \overset{\uparrow}{k=1}$$

Bsp.: mit steigender Flexibilität des Modells (#Nachbarn nimmt ab) sinken die Modellfehler erstmal.

- Ab einem bestimmten  $k$  fängt der Testfehler wieder zu steigen an  $\Rightarrow$  OVERFITTING

⇒ Optimum ungefähr  $k=7$

## Exkurs: geht KNN auch für Regression?



Ein möglicher (optimaler) Prädiktor wäre die Regressionsfunktion

$$f(x) = E(Y | X=x)$$

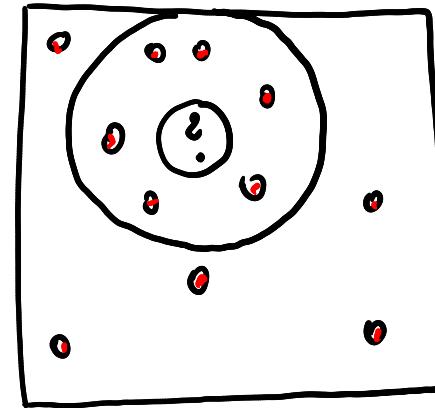
(Y metrisch)

die könnten wir aus den Daten schätzen via:

neues  $x = (x_1, x_2, \dots)$  mit gesuchtem  
 $y$ -Wert?

→ bilden die Nachbarschaft  $W^t(x)$

liefert  $k$  bekannte  
 $\hat{y}$ -Werte



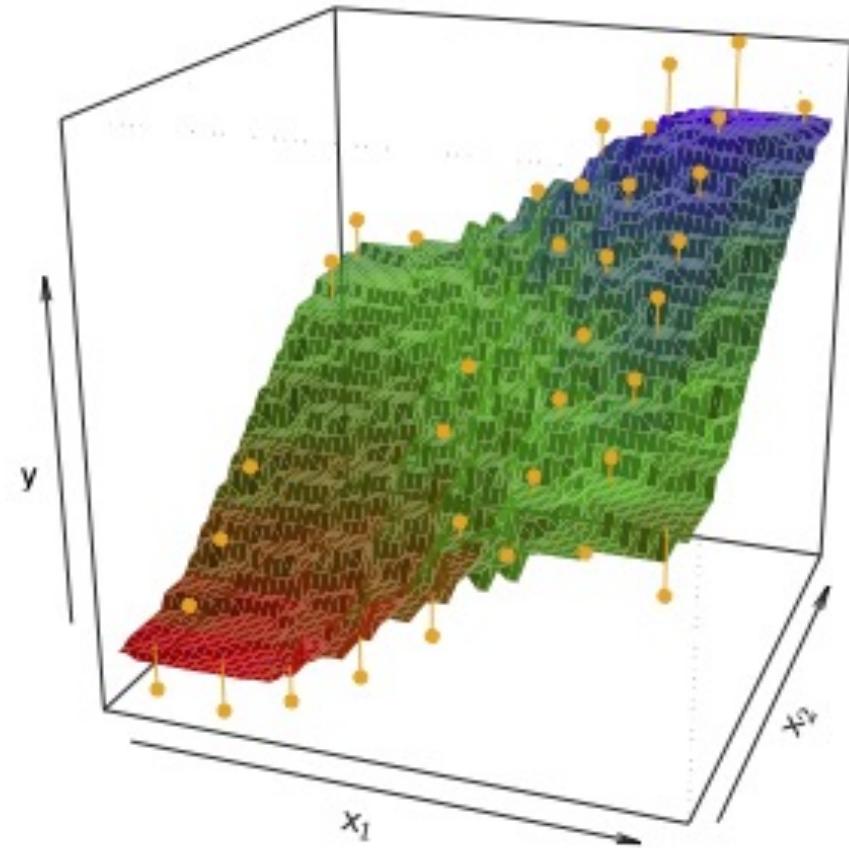
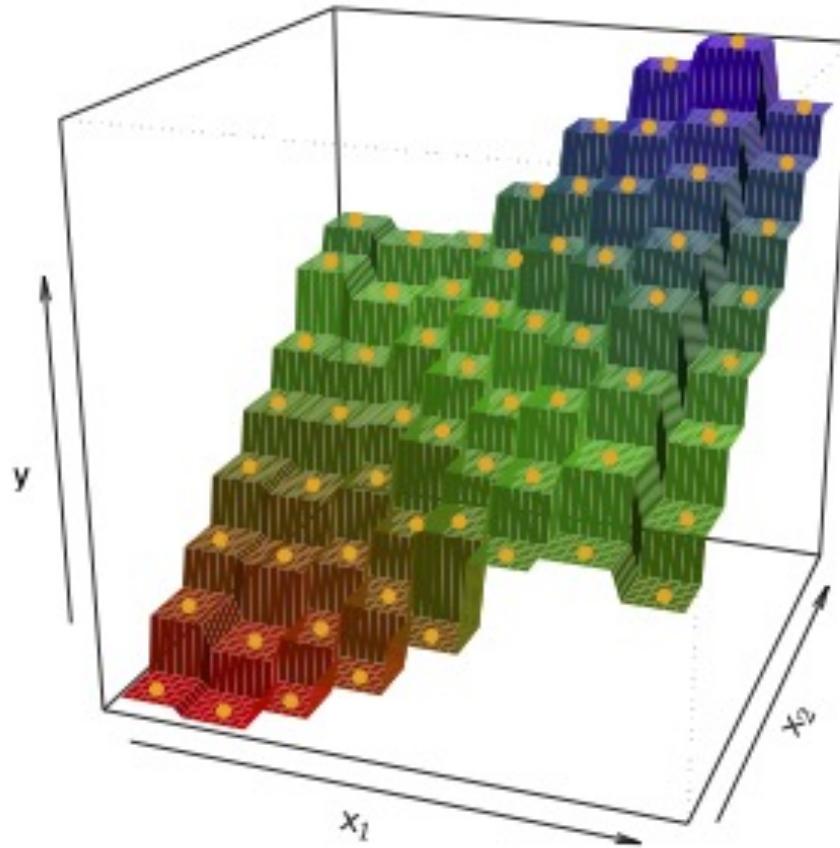
⇒ Schätzung für den Erwartungswert ist das arithmetische Mittel der  $k \hat{y}$ -Werte

$$\Rightarrow \hat{y} = E(Y | X=x) \approx \frac{1}{k} \sum_{\hat{y} \in W^t(x)} \hat{y}$$

⇒ Stufenfunktion

Bsp.: (Tibshirani)

2 Features  $x_1, x_2$



→ Das Ergebnis ist in jedem Fall keine stetige Funktion  $f(x_1, x_2)$  sondern eine Stufenfunktion

## Diskussion: pro & contra KNN

163

### PRO:

- KNN ist nicht-parametrisch  
⇒ keine a priori Hypothese über die Struktur der Aufgabe nötig
- Algorithmus ist leicht verständlich und logisch
- leicht zu implementieren (zumindest in der Theorie)

### CONTRA:

- der "Curse of Dimensionality" schlägt voll zu!  
⇒ Trainingsdatensatz muss sehr groß sein für hochdim. Probleme (einhd. Abstände werden sehr groß)
- Interpretierbarkeit ist schlecht: wir können i.A. nicht sagen wie sich ein Feature auf das Target auswirkt.

## 2.2.4) Überlegungen zur Bewertung der Performance einer Klassifikation

164

Schon gesehen: Die Error-Rate (oder Hit Rate) ist eine Möglichkeit, die Performance zu bewerten:

$$\frac{1}{n} \sum_{i=1}^n (y_i \neq \hat{y}_i) = \frac{\text{Anzahl falscher Zuordnungen}}{\text{Anzahl n der Datenpunkte}}$$



Wie gut ist das als Metrik?

Bsp.:

2 Klassen  
1 (10x)  
0 (990x)

naives Modell = immer 0

$$\Rightarrow \text{ErrorRate} = \frac{10}{1000} \stackrel{!}{=} 1\%$$

ziemlich klein!!  
Trotzdem ist das Modell ungenig 😞



Betrachte die Performance spezifisch in jeder Klasse

besonders wichtig bei  
"unbalanced classes" -  
Problemen !!

## Def.: Confusion Matrix

Die Darstellung

- korrekt
- falsch

		truth					
			1	2	3	4	5
pred	1						
	2						
3	1						
	2						
4	1						
	2						
5	1						
	2						

die jedem Kästchen die  
relativen (absoluten) Häufigkeiten  
der Kombis (pred, truth)  
zuordnet heißt

CONFUSION - MATRIX

Bsp.:

	1	0	6
1	7	0	6
2	2	10	1
3	7	2	5
pred.	1	2	3

korrekt ✓

7x sagt das Modell 1 und es  
war auch eine 1

⚠ falsch !!

6x sagt das Modell 3 obwohl  
es im Wahrheit eine 1 war 😞 ↴ ↴

➡ Mit Hilfe der Confusion-Matrix

kann man die vorhandenen Fehl-  
klassifizierungen genauer analysieren:

- macht eine Gruppe besonders oft Probleme?
- trifft man auch bei unterrepräsentierten Klassen?
- Wie siehts insgesamt aus?

im Fall einer binären Klassifikation:

167

		prediction	
		+	-
truth	+	true positive tp	false negative fn
	-	false positive fp	true negative tn

korrekt positiv

zugeordnet

= true positive

korrekt negativ

zugeordnet

= true negative

... das führt zu folgender Idee

## Def: Accuracy

Es seien  $tp = \text{true positives}$  sowie  $fp = \text{false positives}$   
 $tn = \text{true negatives}$   $fn = \text{false negatives}$

fälschlicherweise  $\oplus$   
 $\downarrow$  statt  $\ominus$

$\uparrow$   
 Fälschlicherweise  $\ominus$   
 statt  $\oplus$

Dann hast

$$\frac{\text{Zahl korrekt}}{\text{Zahl gesamt}} = \frac{tp + tn}{tp + tn + fp + fn}$$

ACCURACY

und

$$\frac{\text{Zahl falsch}}{\text{Zahl gesamt}} = \frac{fp + fn}{tp + tn + fp + fn}$$

entspricht der  
Fehler Rate

Es gilt:

$$\text{Error Rate} = 1 - \text{Accuracy}$$

← klar ↵

Um jetzt die Performance innerhalb der einzelnen Klasse zu bewerten kann man ähnlich argumentieren:

		prediction	
		+	-
truth	+	true positive tp	false negative fn
	-	false positive fp	true negative tn

↳ Anzahl aller tatsächlichen  $\oplus$

$$\text{recall} = \frac{tp}{tp + fn} \quad (\leftarrow \text{True Positive Rate})$$

" Wieviel der echten  $\oplus$  wurden vom Modell erkannt? "

Anzahl aller modellierten  $\oplus$

$$\Rightarrow \text{precision} = \frac{tp}{tp + fp}$$

" Wieviel der modellierten  $\oplus$  waren tatsächlich echte  $\oplus$ ? "



ZIEL Optimiere idealerweise sowohl precision als auch recall !!

... es gibt aber auch Fälle, in denen eins von beiden mehr Gewicht hat

Bsp.: • Spam Mail Classifier

Es ist vernünftig schlimmer, wenn ein wichtiges Mail verloren geht als wenn 1 Spam-Mail mal durchkommt!

⇒ false Spam ( $f_p$ ) sollte möglichst klein sein

⇒ (große) precision:  $\frac{t_p}{t_p + f_p}$  wäre schön... ( $P \hat{=} \text{Merkhilfe } \text{😊}$ )

• Corona Test

Ein positiver Fall sollte möglichst nicht "negativ" klassifiziert werden.

⇒ false negative Corona ( $f_n$ ) sollte möglichst klein sein

⇒ große recall:  $\frac{t_p}{t_p + f_n}$  wäre schön...

## Erweiterung für mehrere Klassen:

	1	2	3	4
1	tp <sub>1</sub>	fp <sub>2</sub>	fp <sub>3</sub>	fp <sub>4</sub>
2	fp <sub>1</sub>	tp <sub>2</sub>		
3			tp <sub>3</sub>	
4				tp <sub>4</sub>
pred.	1	2	3	4

$$\frac{tp_1}{tp_1 + fp_2 + fp_3 + fp_4} = \text{precision}_1$$

für alle Klassen zu berechnen

$$\frac{tp_1}{tp_1 + fp_1 + fp_2 + fp_3} = \text{recall}_1$$

für alle Klassen zu berechnen

### GESAMT:

- Micro-average Precision:  $\frac{\text{total tp}}{\text{total tp} + \text{total fp}}$
- Macro-average Precision:  $\frac{1}{\# \text{classes}} \sum_{i=1}^{\# \text{classes}} \text{prec}_i$
- Weighted average Precision: gewichtete prec; im average mit KlassengröÙe

Analog: recall

# Nochmal zur binären Klassifikation

Bsp.: Spam-Filter

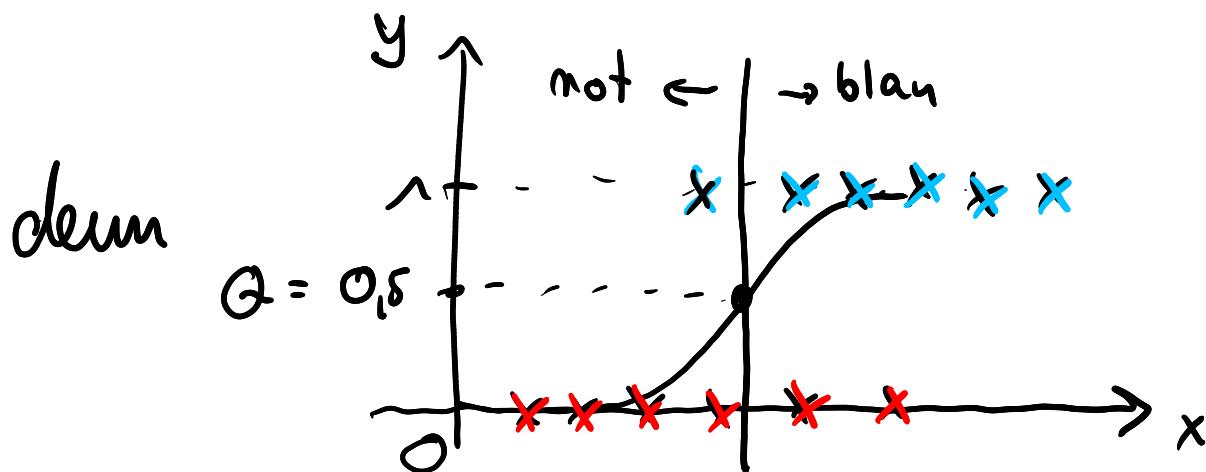
$$Y = \begin{cases} ? \text{ Spam} \\ \rightarrow \text{No Spam} \end{cases}$$

} Angenommen wir machen logistische Regression, dann entscheiden wir via

$$P(Y=1 | X=x) > 0,5 \Rightarrow Y=1$$

hängt vom  $\Rightarrow$   
Threshold 0.5 ab

		prediction	
		+	-
truth	+	true positive ✓ tp	false negative ✗ fn
	-	false positive ✗ fp	true negative ✓ tn



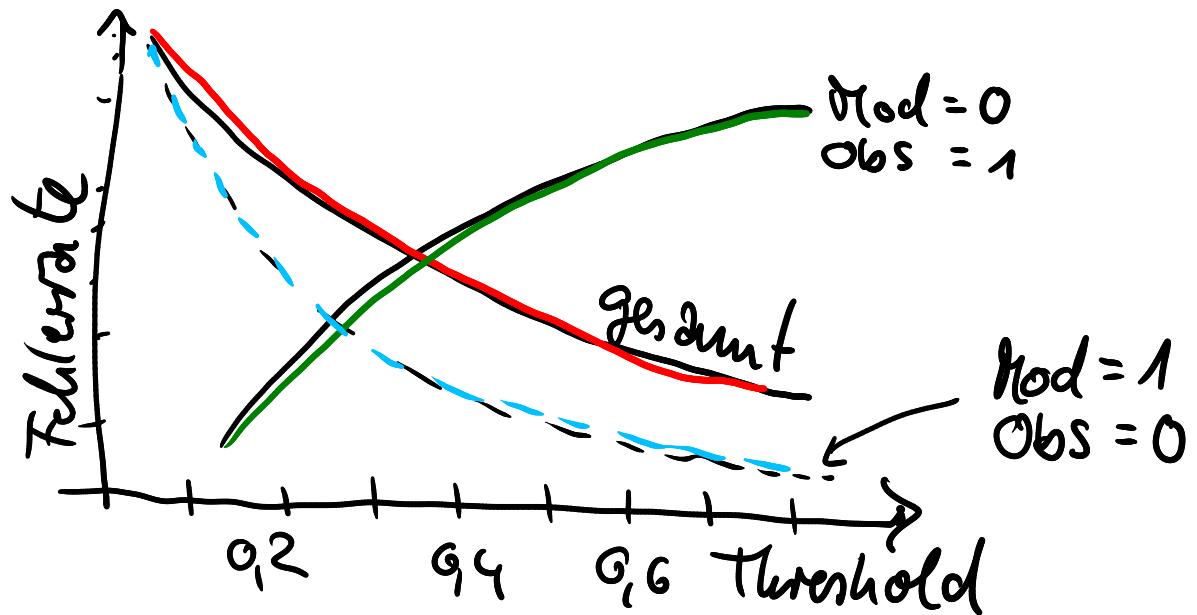
Hat man z.B. eingesehen, dass zu viele Fehler der Art "0 obwohl eigentlich 1" passieren, dann könnte man den Threshold ja einfach kleiner machen!

Z.B.  
⇒

$$P(Y=1 | X=x) > 0,2 \Rightarrow Y=1$$

← dann steigt aber leider der 2. Fehler!

DILEMMA



 IDEE

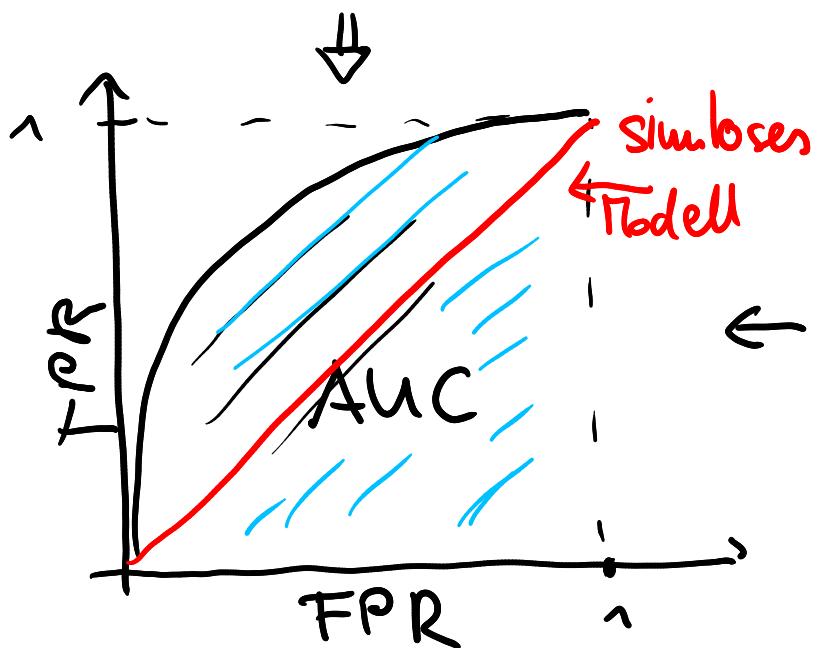
Zeichne die False Positive Rate = Anteil der Oer, die falsch als 1 klass. werden

gegen die True Positive Rate = Anteil der Oer, die korrekt als 1 klass. werden

Bleiu ist gut  
gros ist gut!

für verschiedene Thresholds.

⇒ ROC-Curve (Receiver Operating Characteristics)



$\text{AUC} = \text{Area Under Curve}$   
= Fläche unter der ROC-Curve

## INTERPRETATION

Ein Modell mit perfekter Trennschärfe hätte eine ROC-Curve, die durch  $(0; 1)$  geht

$\Rightarrow$  TPR sehr groß, FPR sehr klein

ALSO: je steiler die ROC, desto besser!

Die AUC kann als Summary für die Trennschärfe interpretiert werden (binäres Modell).

ALSO: Je näher die AUC bei 1 liegt desto besser ...