Data Science 1

Vorlesung 7 - Intro

Prof. Dr. Christian Bockermann

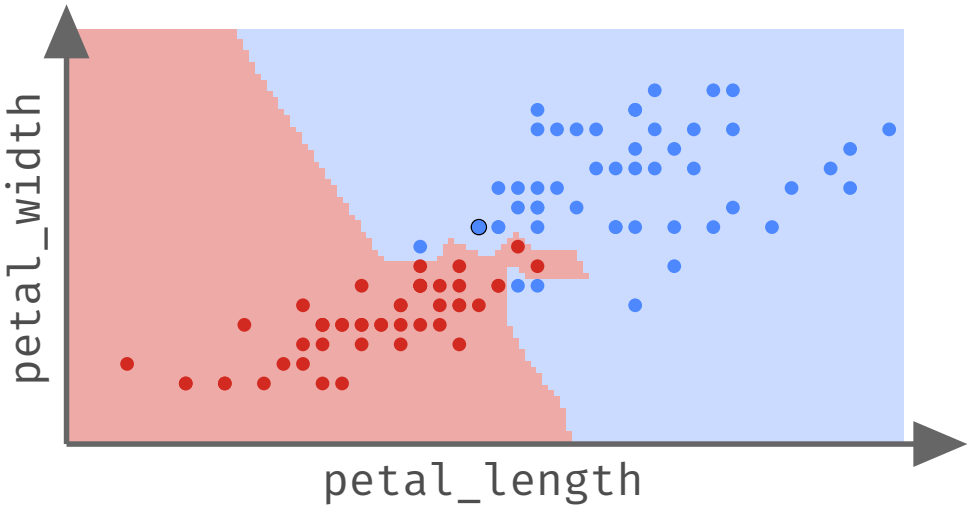
Hochschule Bochum

Wintersemester 2021/2022

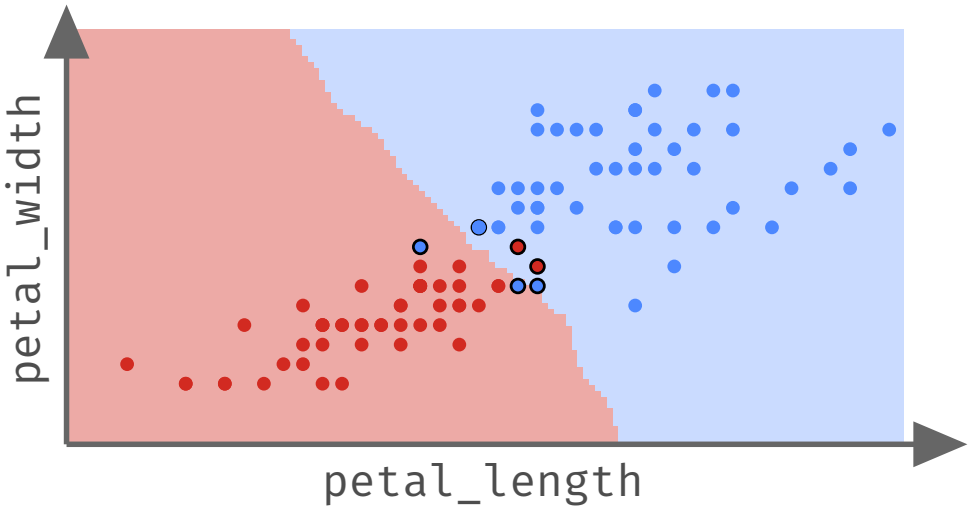
Was geschah zuletzt?

- Instanzbasiertes Lernen über Ähnlichkeit
- Distanz-Funktion auf Beispielen (eukl. Distanz)
- Normalisierung von Daten (Min/Max-, z-Normalisierung)
- k-NN als Vorhersagemodell

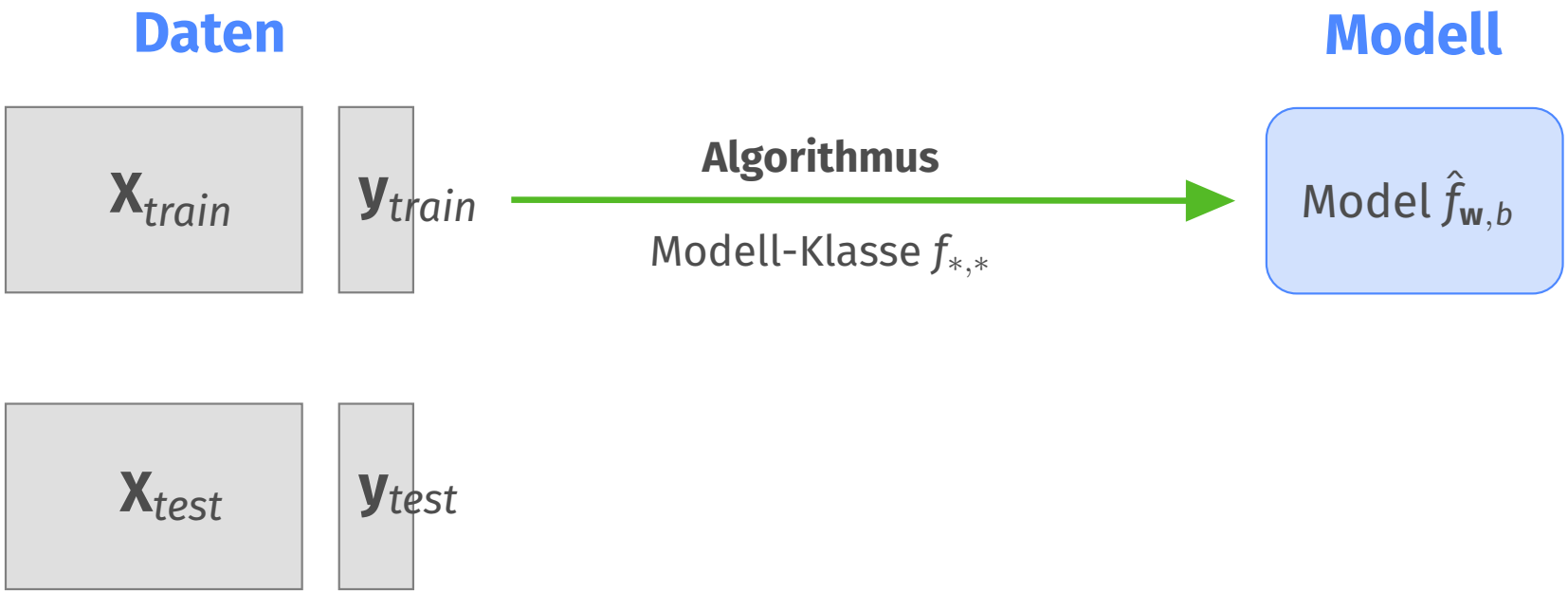
k=1



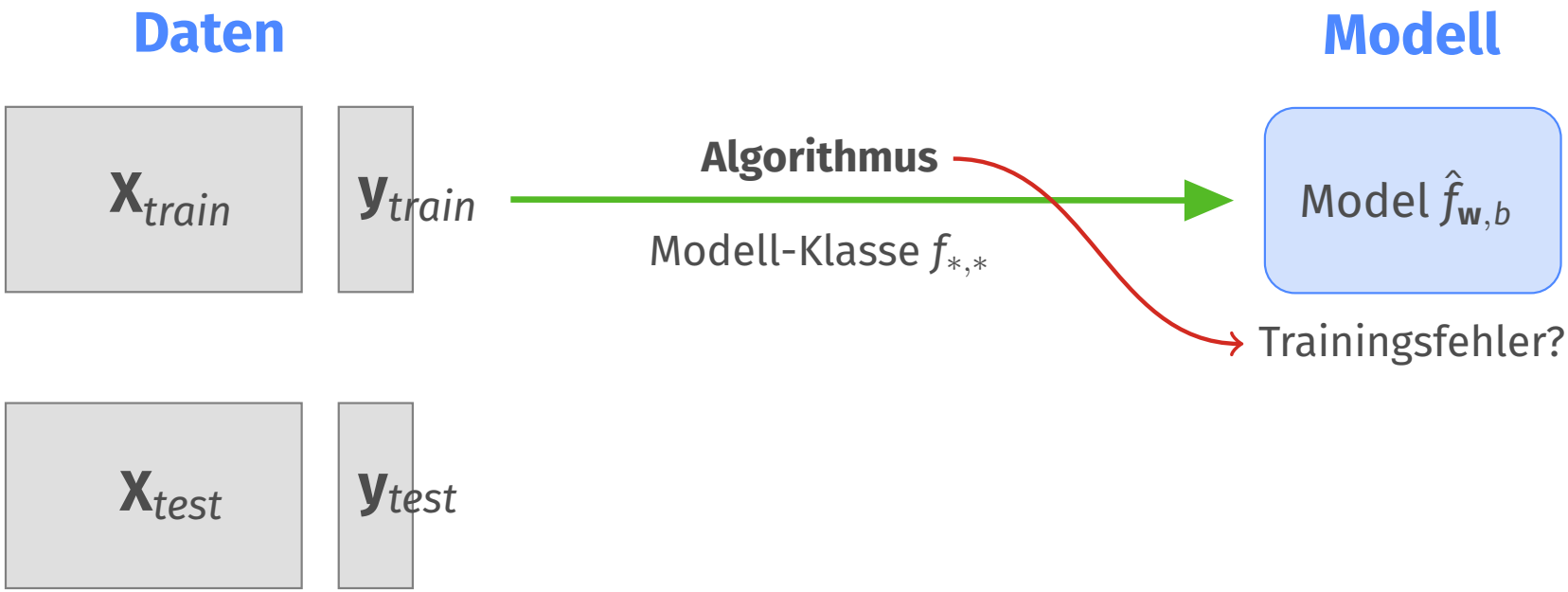
k=16



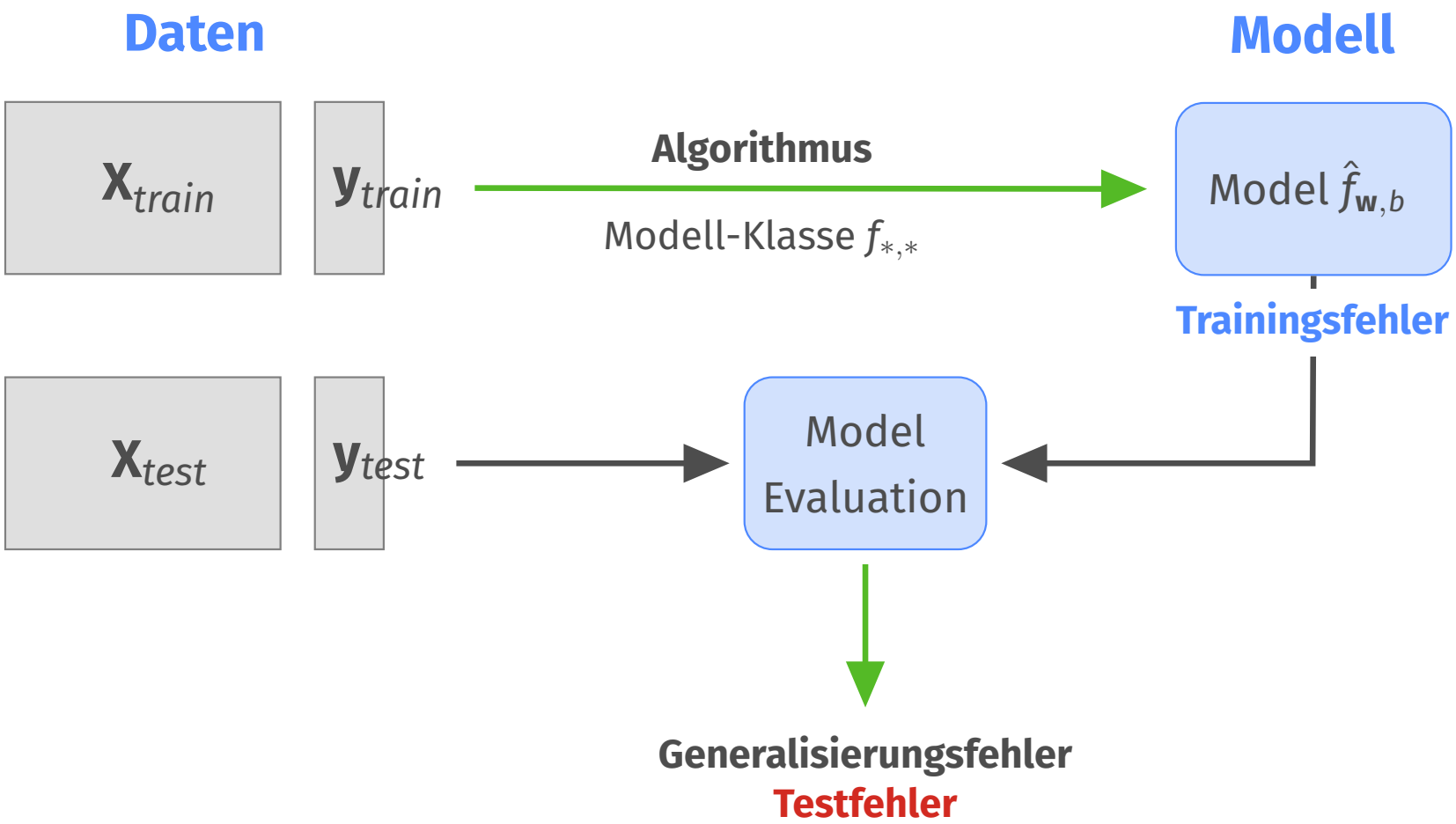
Bewertung der Modell-Güte – Generalisierungsfehler



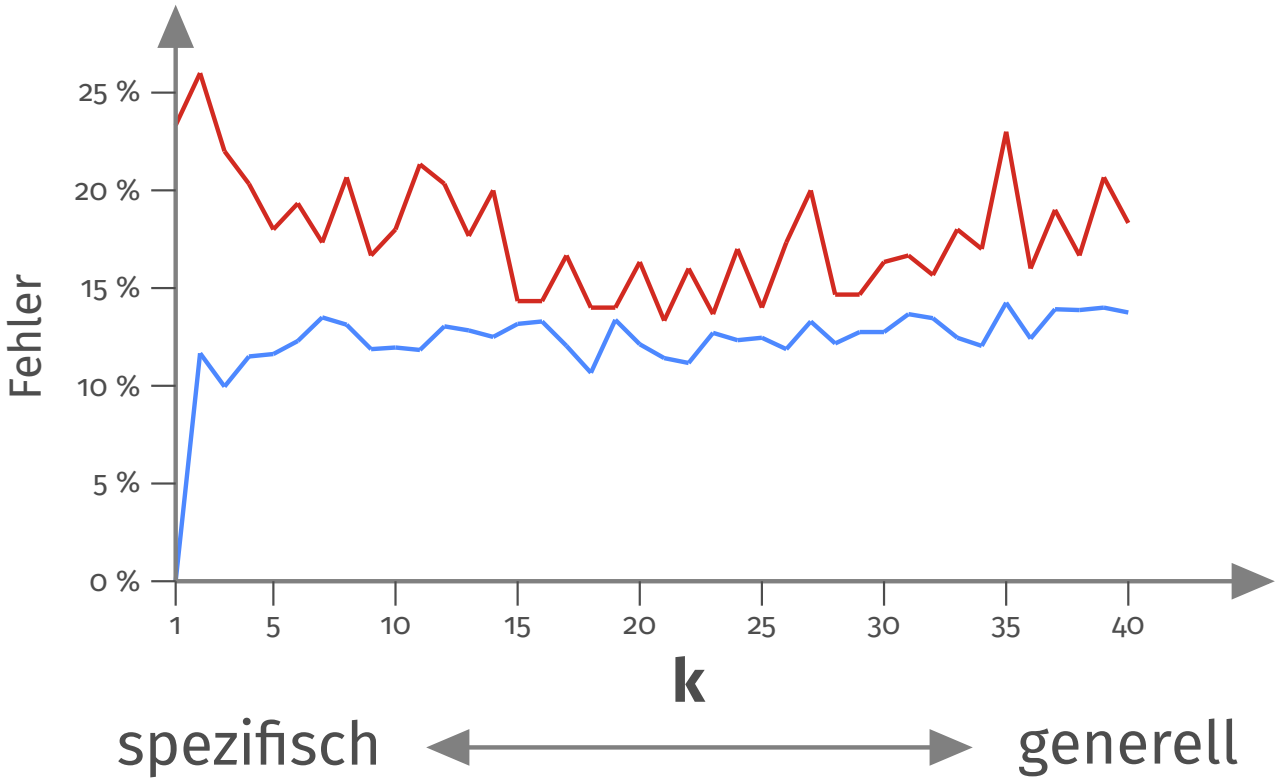
Bewertung der Modell-Güte – Generalisierungsfehler



Bewertung der Modell-Güte – Generalisierungsfehler



Trainings- und Test-Fehler auf generiertem Datensatz (k-NN)



Overfitting

“Das Modell passt nur zu den Trainingsdaten.”

Overfitting

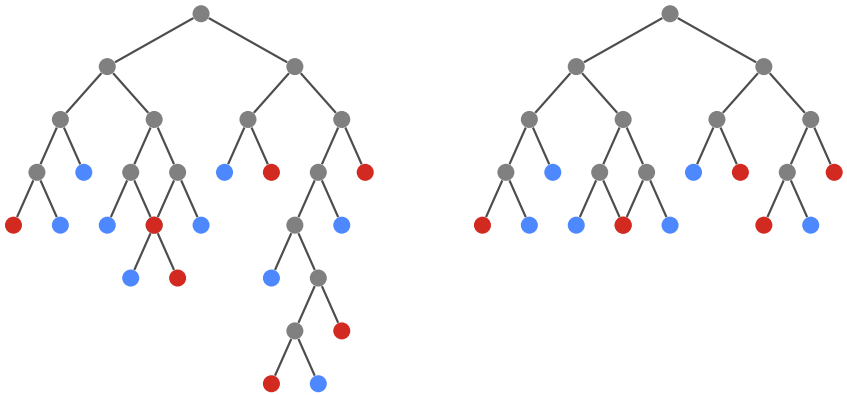
"Das Modell passt nur zu den Trainingsdaten."

	Trainingsfehler **klein**	Trainingsfehler **groß**
Testfehler **klein**	Das sieht gut aus!	
Testfehler **groß**	**Overfitting!**	Das Modell lernt nicht!?

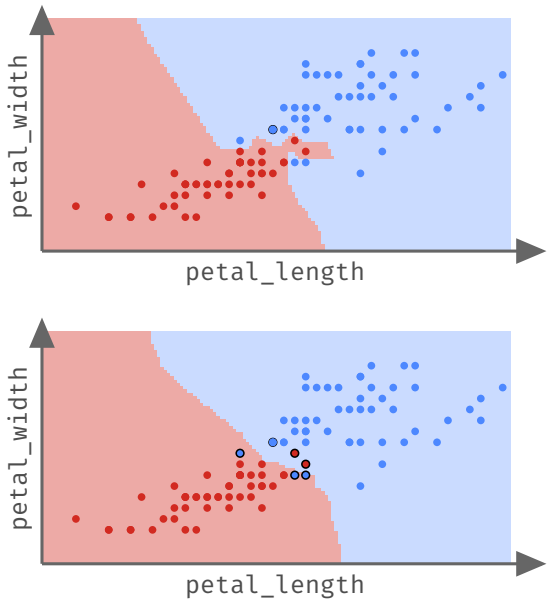
Overfitting - zu spezifisches Modell

- Modell zu sehr an die Trainingsdaten angepasst
- Vorhersage auf unbekannten Daten schlechter
- Modellkomplexität begrenzen (generelleres Modell)

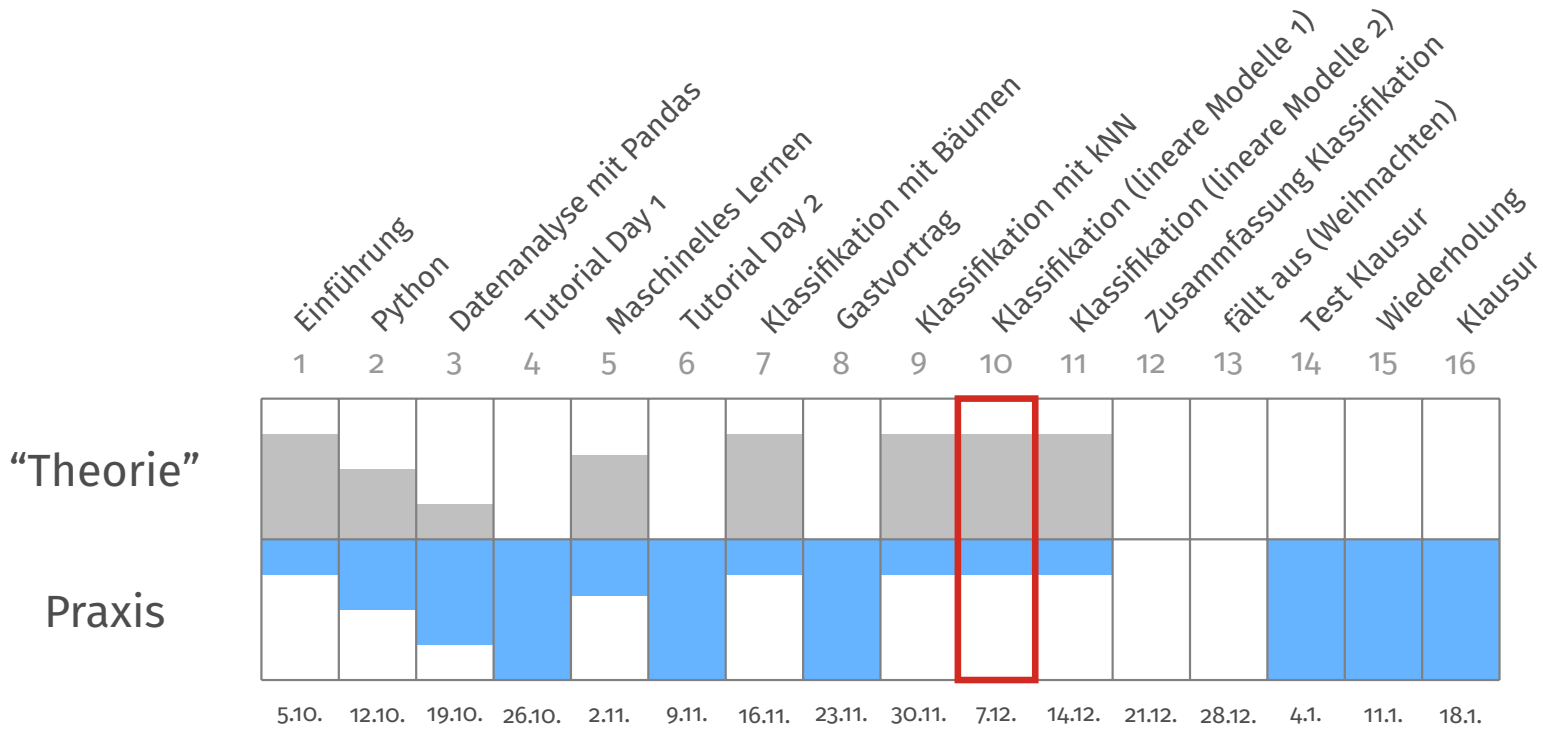
Tiefe bei Bäumen beschränken



k bei k-NN erhöhen



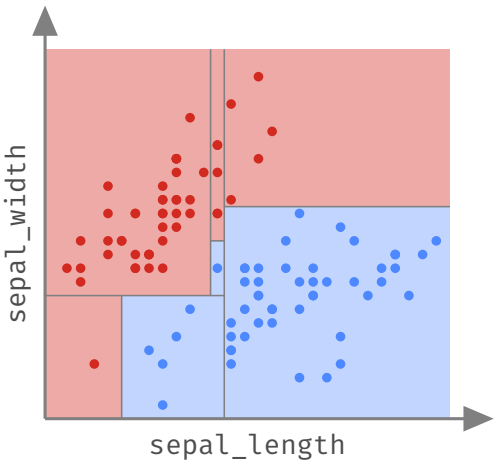
Wo sind wir heute (Vorlesung 7) ?



Worum geht es im Foliensatz 7?

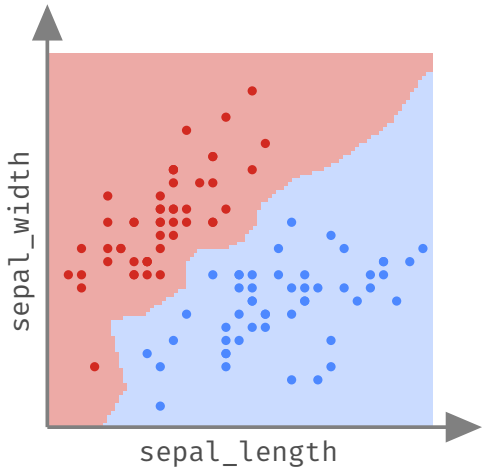
1. Daten im Vektorraum
2. Lineare Modelle
3. Ein einfaches lineares Modell
4. Die Maximum-Margin Idee (SVM)

Entscheidungsbäume, nächste Nachbarn



Entscheidungsbaum

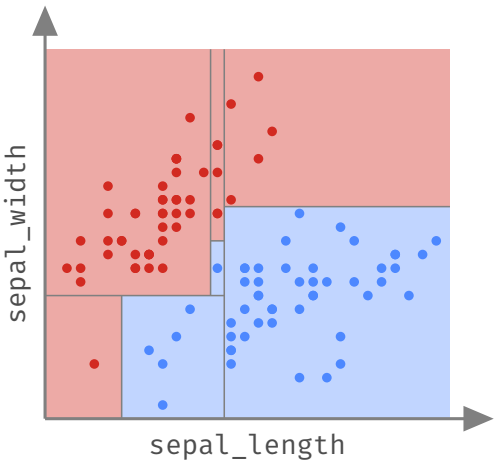
Trennung nach einzelnen Attributen, achsenparallel



k-nächste Nachbarn

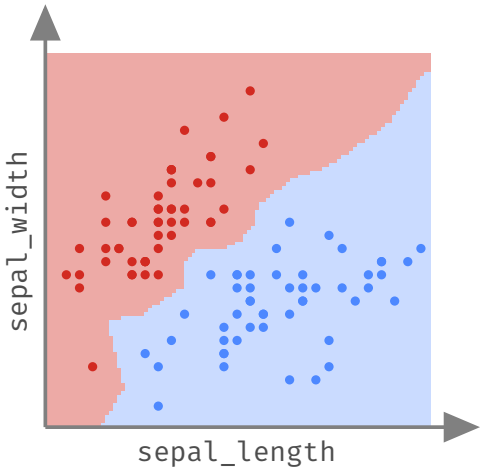
Trennung in Regionen, nach Distanz (Berechnung über alle Attribute)

Entscheidungsbäume, nächste Nachbarn und lineare Modelle



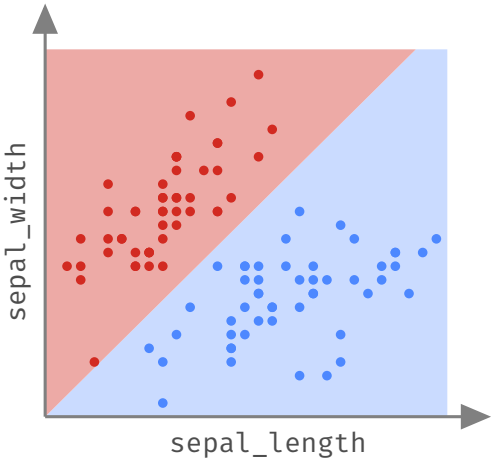
Entscheidungsbaum

Trennung nach einzelnen Attributen, achsenparallel



k-nächste Nachbarn

Trennung in Regionen, nach Distanz (Berechnung über alle Attribute)



Lineare Modelle

Trennung mit linearer Funktion über alle Attribute

Vektorraum: Iris-Daten im Vektorraum $\mathbb{R}^4$

sepal_length	sepal_width	petal_length	petal_width
4.700	3.200	1.300	0.200
6	2.200	4	1
4.600	3.100	1.500	0.200
7.600	3	6.600	2.100
6.300	2.900	5.600	1.800
5.400	3.900	1.700	0.400

$$\mathbf{x}_3 = \begin{pmatrix} 4.6 \\ 3.1 \\ 1.5 \\ 0.2 \end{pmatrix}$$

Vektor-Darstellung der
Zeile 3 aus dem Datensatz

Vektorraum: Iris-Daten im Vektorraum $\mathbb{R}^2$

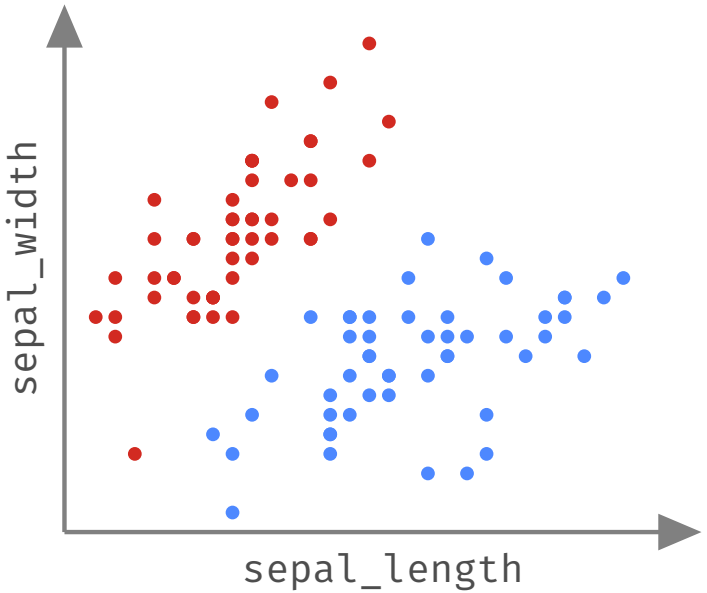
sepal_length	sepal_width
4.700	3.200
6	2.200
4.600	3.100
7.600	3
6.300	2.900
5.400	3.900

$$\mathbf{x}_3 = \begin{pmatrix} 4.6 \\ 3.1 \end{pmatrix}$$

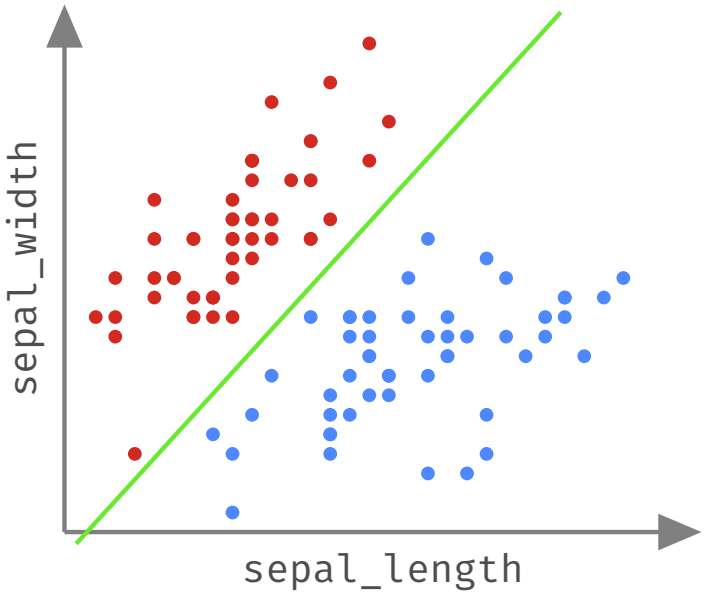
Vektor-Darstellung der
Zeile 3 aus dem Datensatz

Betrachten wir vereinfacht nur 2 Attribute!

Plot der beiden Attribute der Iris-Daten:

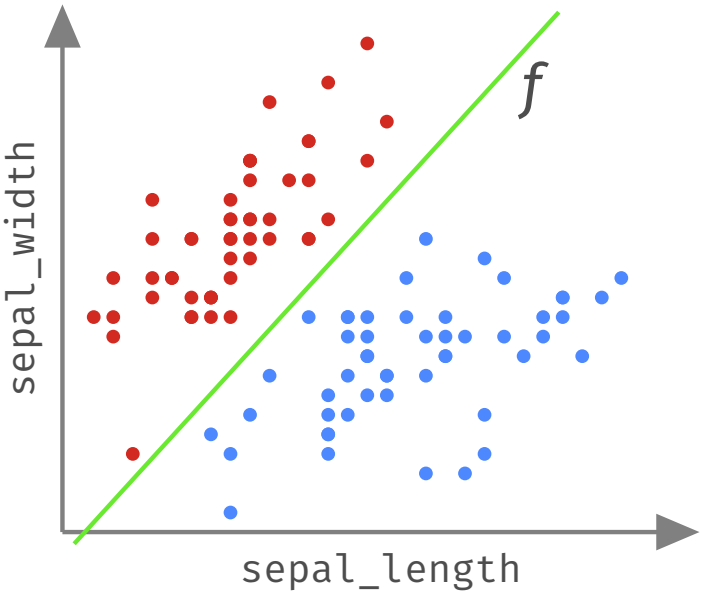


Plot der beiden Attribute der Iris-Daten:



Intuitiv lassen sich die beiden Klassen trennen.

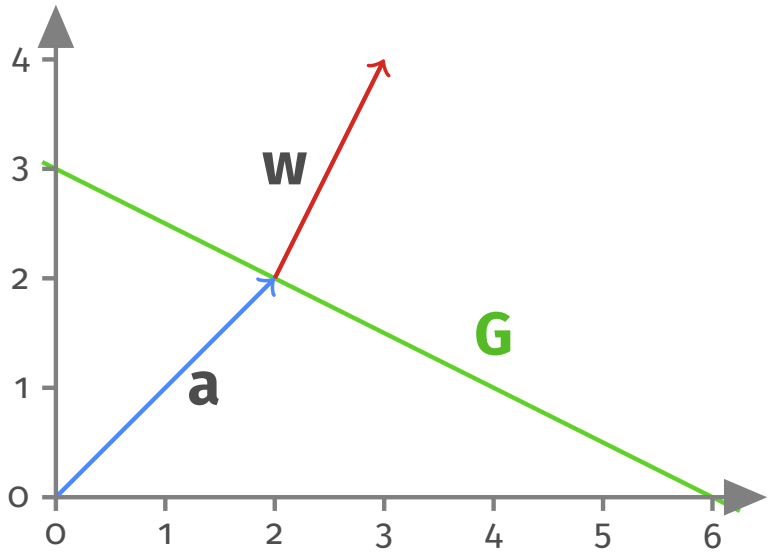
Idee: Daten mit einer Geraden trennen



Geradengleichung (Schule) im 2-dimensionalen Raum ($\mathbb{R}^2$):

$$f(x) = b \cdot x + c$$

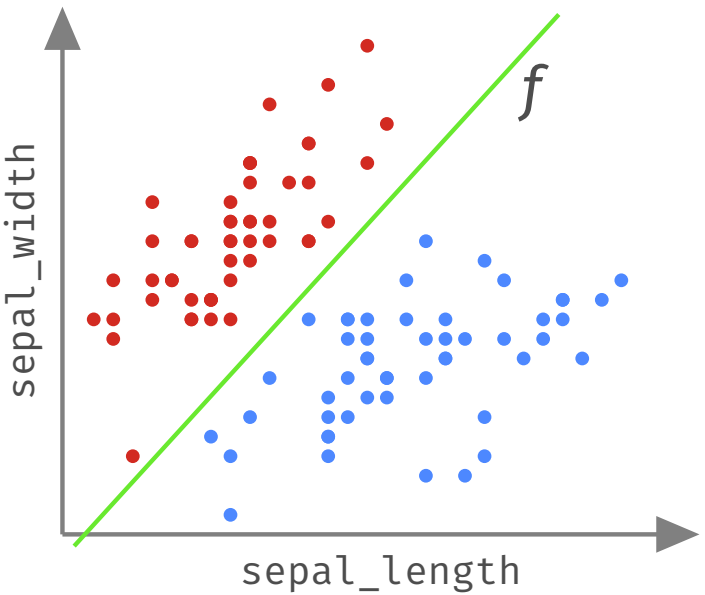
Beispiel: Gerade im $\mathbb{R}^2$



Gerade ist definiert durch Stützvektor $\vec{a} = \begin{pmatrix} 2 \\ 2 \end{pmatrix}$ und Normalenvektor $\vec{w} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$

$$\mathbf{G} = \left\{ (x, y) \in \mathbb{R}^2 \mid 1 \cdot x + 2 \cdot y = 6 \right\}$$

Beispiel: Gerade im $\mathbb{R}^2$



Wir müssen also nur die richtigen **a** und **w** finden!

Allgemeine Form für Hyperebenen im $\mathbb{R}^d$

Hyperebene H definiert durch

$$H = \left\{ \mathbf{x} \in \mathbb{R}^d \mid \mathbf{w}^T\mathbf{x} + b = 0 \right\}$$

Allgemeine Form für Hyperebenen im $\mathbb{R}^d$

Hyperebene H definiert durch

$$H = \left\{ \mathbf{x} \in \mathbb{R}^d \mid \mathbf{w}^T\mathbf{x} + b = 0 \right\}$$

Parametrisierung von f durch $\mathbf{x}$ und b:

$$f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$$

Modell-Training ist damit die Auswahl von w und b!

Ein einfaches lineares Modell

Wie kommen wir nun zu einer trennenden Ebene?

Beispiel: Einfacher Algorithmus über Klassenmittelpunkte

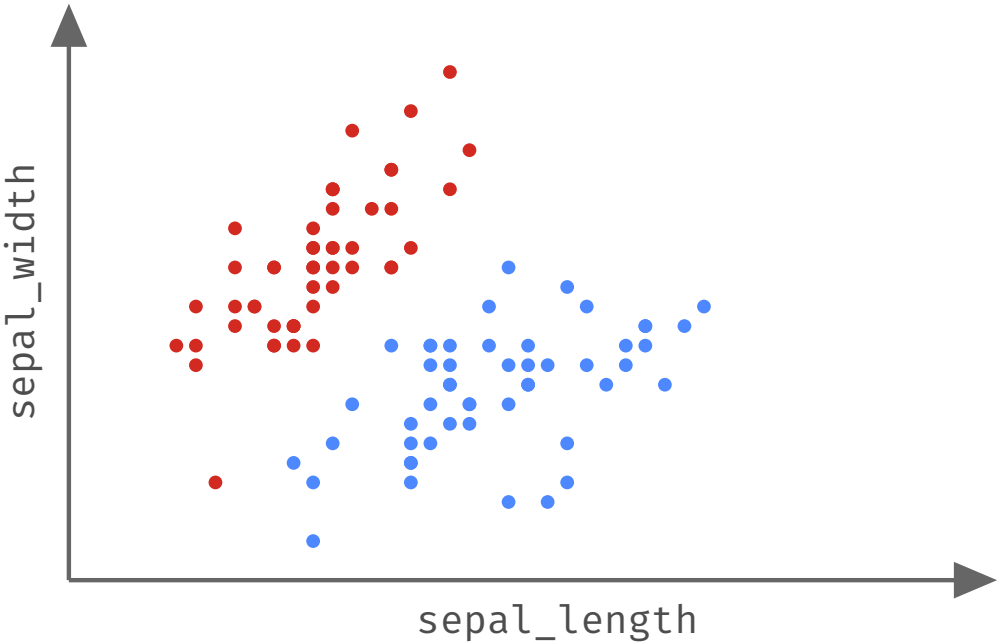
Es sei hier $\mathbf{X}_{[K]}$ die Menge der Daten der Klasse K und $|\mathbf{X}_{[K]}|$ die Anzahl der Punkte in der Menge

Die Mittelpunkte der Klassen setosa und versicolor

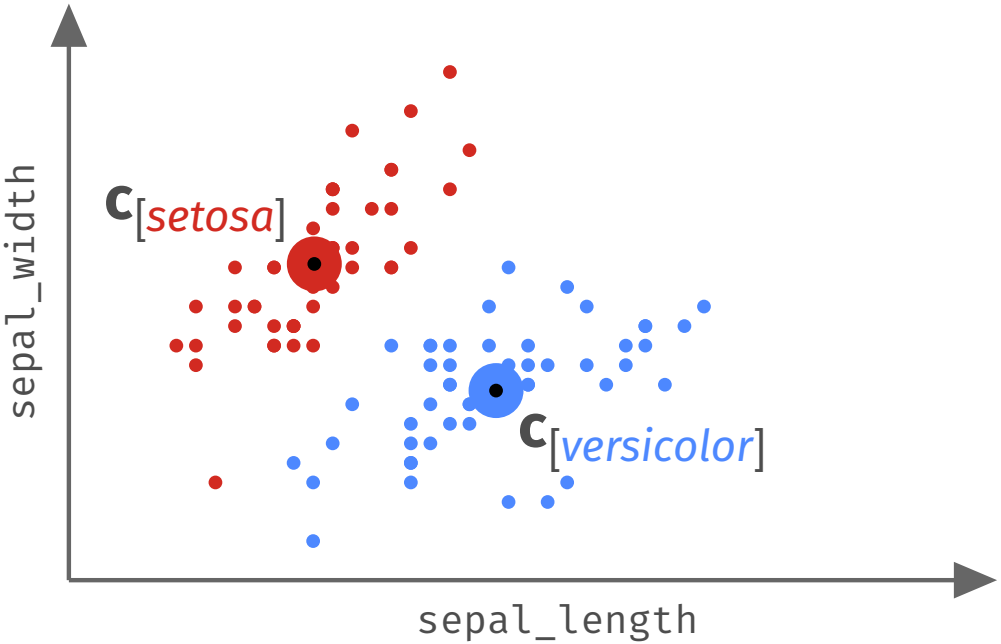
$$\mathbf{c}_{setosa} = \frac{1}{|\mathbf{X}_{[setosa]}|} \sum_{\mathbf{x} \in \mathbf{X}_{[setosa]}} \mathbf{x} \tag{1}$$

$$\mathbf{c}_{versicolor} = \frac{1}{|\mathbf{X}_{[versicolor]}|} \sum_{\mathbf{x} \in \mathbf{X}_{[versicolor]}} \mathbf{x} \tag{2}$$

Wie kommen wir nun zu einer trennenden Ebene?

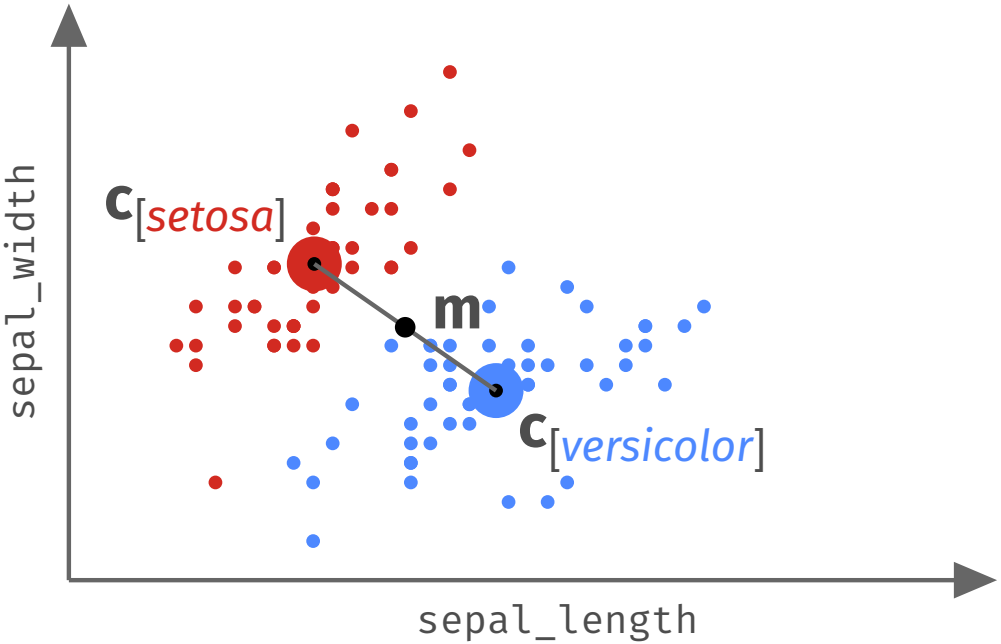


Wie kommen wir nun zu einer trennenden Ebene?



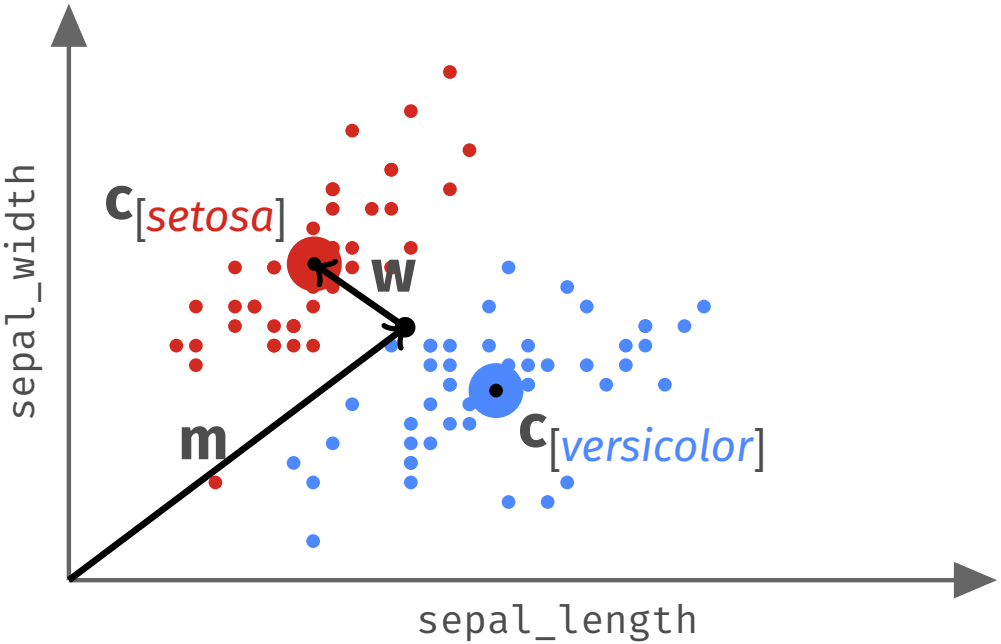
(1.) Berechne die Klassen-Mittelpunkte $\mathbf{c}_{[setosa]}$ und $\mathbf{c}_{[versicolor]}$.

Wie kommen wir nun zu einer trennenden Ebene?



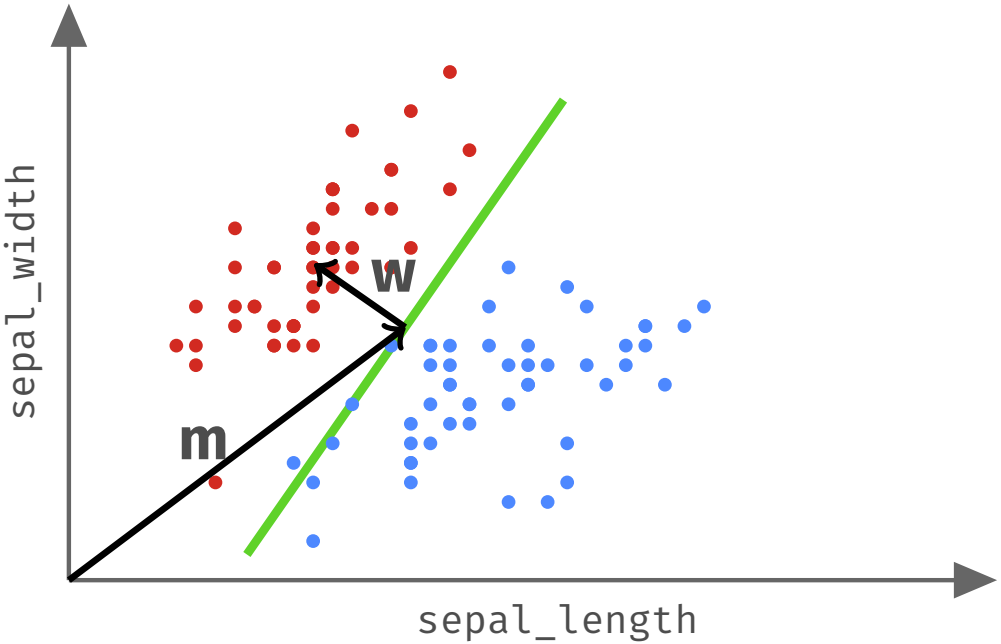
(2.) Bestimme Mittelpunkt $\mathbf{m} = \frac{1}{2}(\mathbf{c}_{[versicolor]} + \mathbf{c}_{[setosa]})$

Wie kommen wir nun zu einer trennenden Ebene?



(3.) Stützvektor $\mathbf{m}$, Normalenvektor $\mathbf{w} = \mathbf{c}_{[setosa]} - \mathbf{m}$

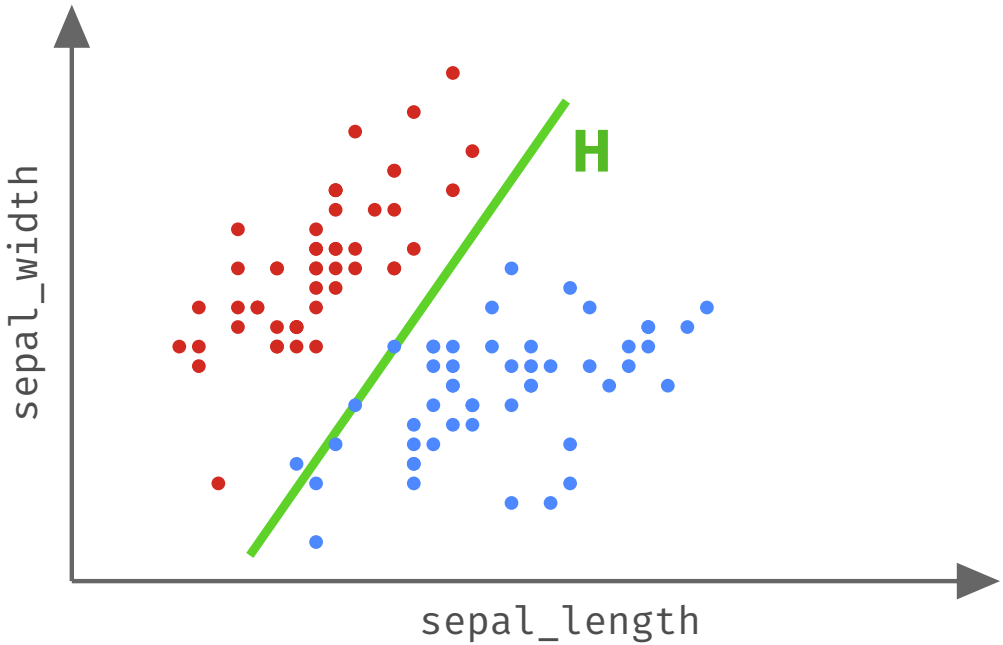
Wie kommen wir nun zu einer trennenden Ebene?



(3.) Ebene mit Stützvektor $\mathbf{m}$, Normalvektor $\mathbf{w}$:

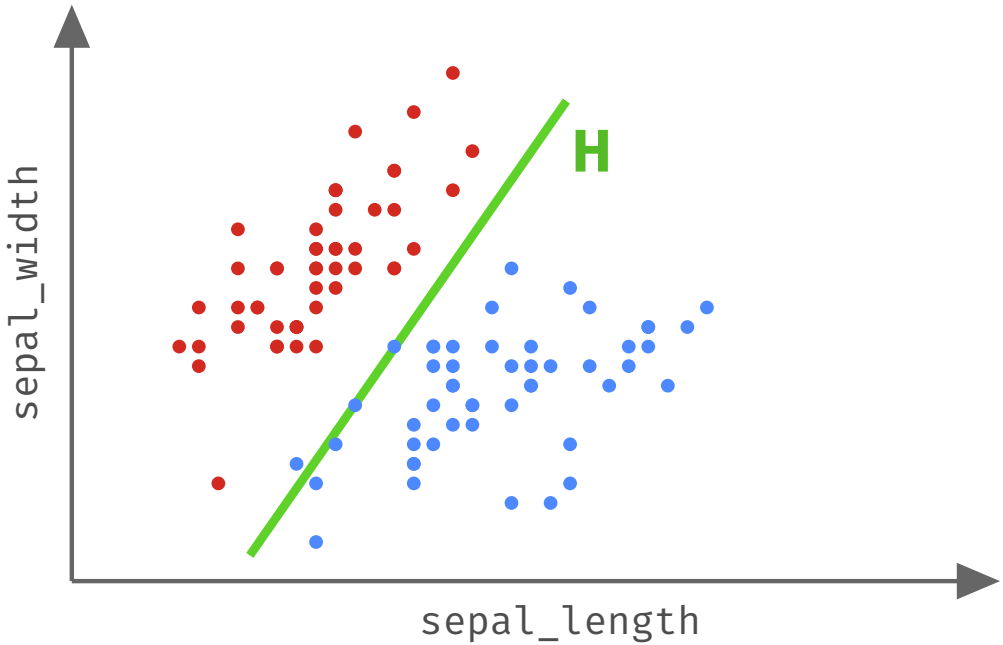
$$\mathbf{w}^T(\mathbf{x} - \mathbf{m}) = 0 \Leftrightarrow \mathbf{w}^T\mathbf{x} - \underbrace{\mathbf{w}^T\mathbf{m}}_{\approx -1.54} = 0$$

Ergebnis des einfachen Verfahrens:



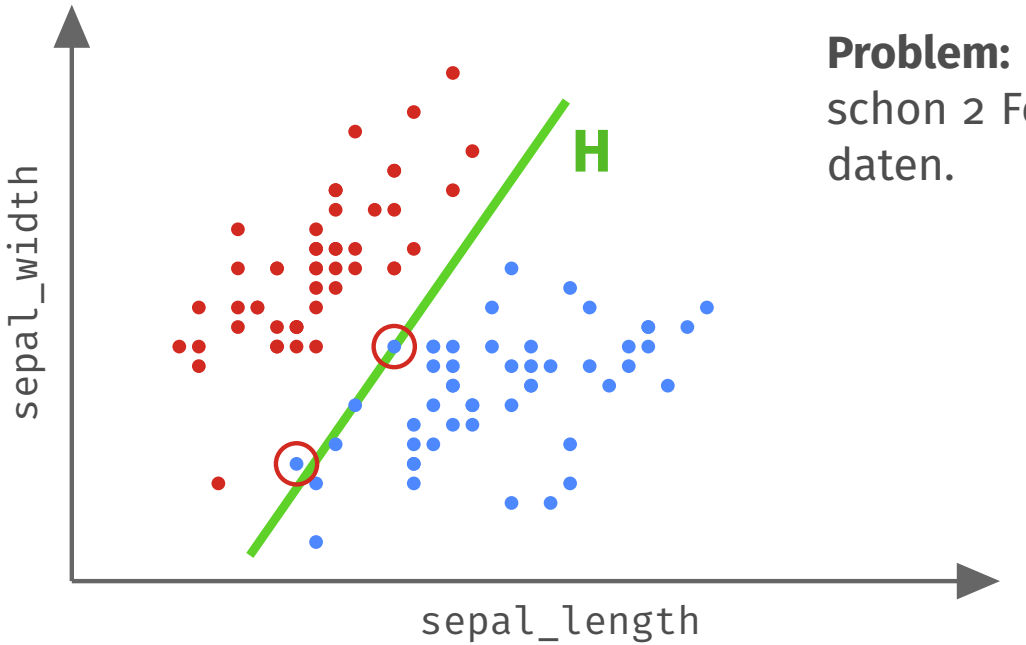
$$\mathbf{H} = \left\{ \mathbf{x} \in \mathbb{R}^2 \mid \mathbf{w}^T \mathbf{x} = -1.54 \right\}, \ \mathbf{w} = \begin{pmatrix} -0.465 \\ 0.324 \end{pmatrix}$$

Ergebnis des einfachen Verfahrens:



$$\mathbf{H} = \left\{ \mathbf{x} \in \mathbb{R}^2 \mid \mathbf{w}^T \mathbf{x} = -1.54 \right\}, \ \mathbf{w} = \begin{pmatrix} -0.465 \\ 0.324 \end{pmatrix}$$

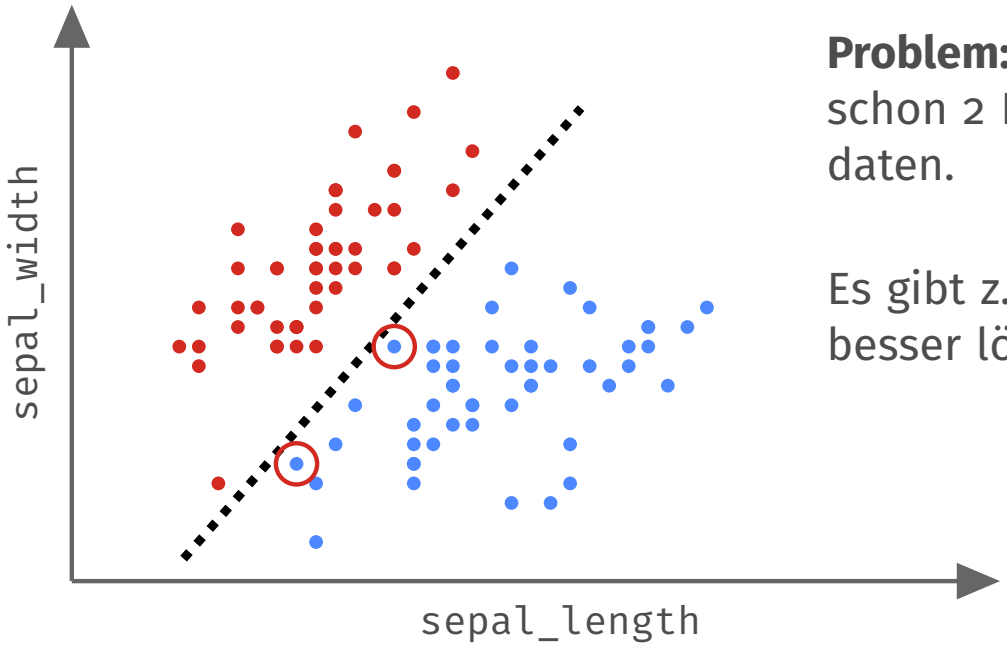
Ergebnis des einfachen Verfahrens:



Problem: Ebene **H** erzeugt schon 2 Fehler auf Trainingsdaten.

$$\mathbf{H} = \left\{ \mathbf{x} \in \mathbb{R}^2 \mid \mathbf{w}^T\mathbf{x} = -1.54 \right\}, \ \mathbf{w} = \begin{pmatrix} -0.465 \\ 0.324 \end{pmatrix}$$

Ergebnis des einfachen Verfahrens:



Problem: Ebene **H** erzeugt schon 2 Fehler auf Trainingsdaten.

Es gibt z.B. Ebene **H***, die das besser löst.

$$\mathbf{H} = \left\{ \mathbf{x} \in \mathbb{R}^2 \mid \mathbf{w}^T \mathbf{x} = -1.54 \right\}, \ \mathbf{w} = \begin{pmatrix} -0.465 \\ 0.324 \end{pmatrix}$$

Die Maximum-Margin Idee (SVM)

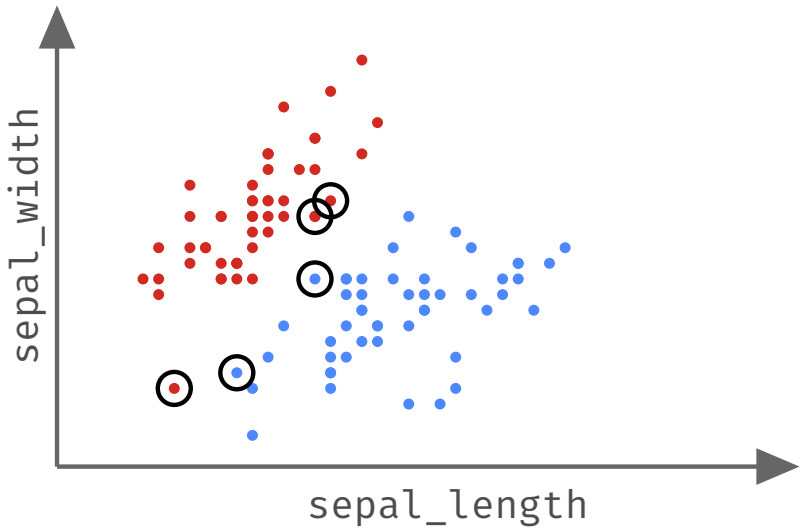
Das einfache Verfahren klappt nicht gut

- Berechnet $\mathbf{w}, b$ ohne Berücksichtigung des Trainingsfehlers
- Bezieht alle Punkte gleich-wichtig in Berechnung von $\mathbf{w}, b$ ein
- Ist nur Heuristik um lineare Modelle zu erklären :-)

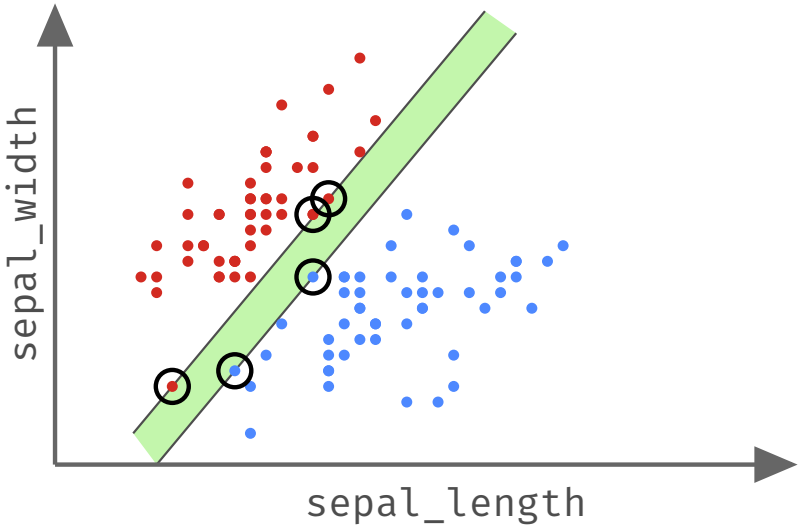
Frage: Wie kommen wir zu guten $\mathbf{w}$ und b?

- Sind alle Datenpunkte gleich wichtig?
-

Betrachten wir die Grenzpunkte der beiden Klassen:

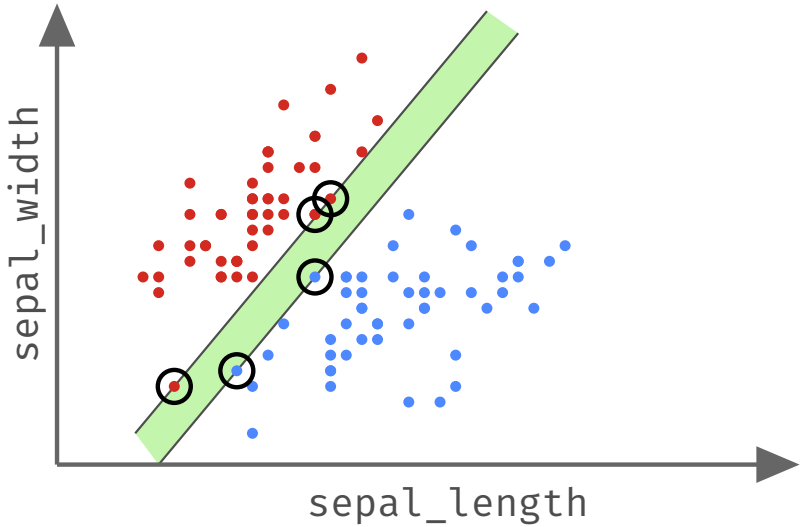


Betrachten wir die Grenzpunkte der beiden Klassen:



- Geraden durch Grenzpunkte erzeugt eine Art *Korridor*
- Grüner Bereich enthält die Ebenen, die fehlerfrei klassifizieren
- Es gibt **unendlich viele** fehlerfreie Ebenen in diesem Bereich

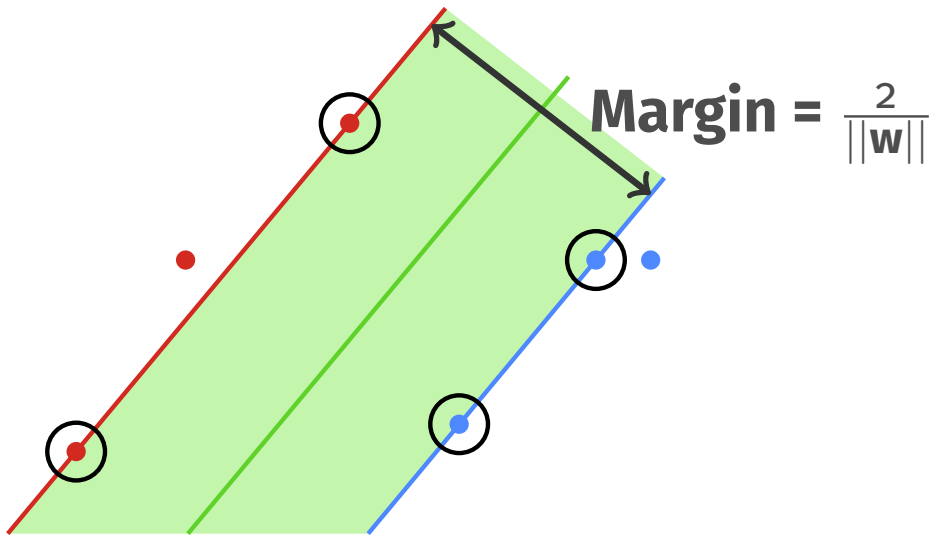
Betrachten wir die Grenzpunkte der beiden Klassen:



- Geraden durch Grenzpunkte erzeugt eine Art *Korridor*
- Grüner Bereich enthält die Ebenen, die fehlerfrei klassifizieren
- Es gibt **unendlich viele** fehlerfreie Ebenen in diesem Bereich

Wir wollen eine Ebene auswählen - welche ist die Beste?

Idee: Wähle Ebene mit größtem Abstand zu beiden Klassen

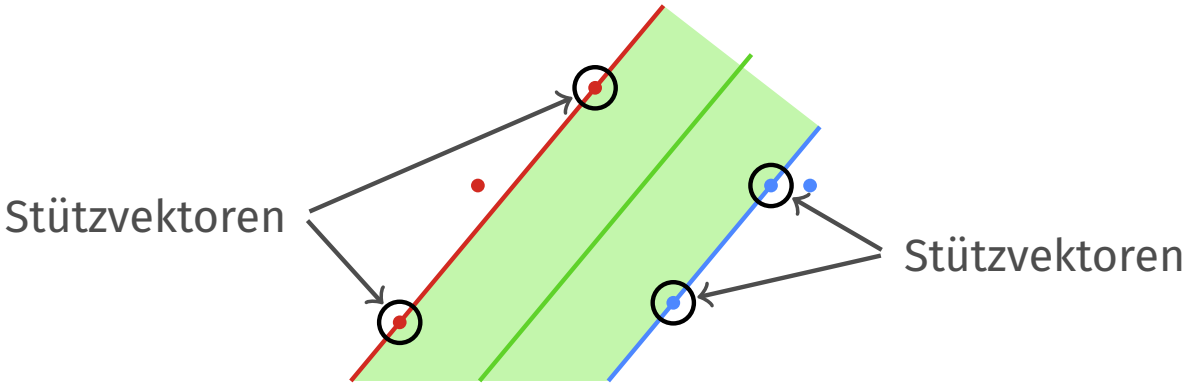


Führt zu Optimierungsproblem (Maximierung des Margin)

Statt $\frac{1}{||\mathbf{w}||}$ zu maximieren, können wir auch $\frac{1}{2}||\mathbf{w}||^2$ minimieren.
(Trick, um ein konvexes Optimierungsproblem zu erhalten)

Klassifikation mit der Support Vector Machine (SVM)

- Support Vector Machine = Stützvektor Methode
- Stützvektoren sind die Punkte, die den Margin definieren



- SVM findet optimale Hyperebene

SciKit Learn enthält SVM Classifier

```
# SVC = Linear Support Vector Classifier
from sklearn.svm import SVC

# Daten Laden, X, y erzeugen

m = SVC(kernel='linear')  # lineares Modell
m.fit(X, y)
```

◀ Probieren Sie es im Notebook aus!

Notebook: Vorlesung/V7-Lineare-SVM.ipynb