

# DATA SCIENCE 1

VORLESUNG 8 - KLASSIFIKATION

PROF. DR. CHRISTIAN BOCKERMANN

HOCHSCHULE BOCHUM

WINTERSEMESTER 2024/2025

1 Lineare Klassifikation

2 Nicht-lineare Separierbarkeit

# Lineare Klassifikation

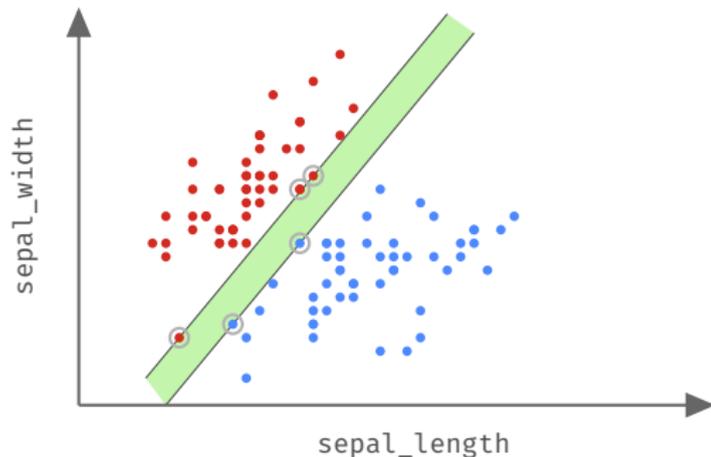
## Lineare Modelle zur Klassifikation

- Beschreibung von trennenden Hyperebenen durch lineare Funktionen
- Suche nach bester Ebene/Funktion durch Optimierung
- Optimierung zielt auf Minimierung des Trainingsfehlers ab

### Beispiel: **Stützvektor-Methode für Klassifikation**

- Such Parameter  $\mathbf{w}$ ,  $\mathbf{b}$  für Ebene
- Optimierung: Maximierung des Abstandes zu beiden Klassen
- Funktioniert gut in hochdimensionalen Räumen

Lineare SVM geht von **linearer Separierbarkeit** aus



**Beste Ebene durch Lösung eines Optimierungsproblems**

## Das SVM Optimierungsproblem

- SVM nutzt Optimierungsproblem um beste Ebene zu finden
- Vorgehen geht von [linearer Separierbarkeit](#) aus
- [Lösung](#) = Ebene mit maximalem Abstand

Optimierungsproblem mit Nebenbedingungen

$$\arg \min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2 \quad \text{mit} \quad (\mathbf{w}^T \mathbf{x}_i + b) y_i \geq 1$$

## Das SVM Optimierungsproblem

- SVM nutzt Optimierungsproblem um beste Ebene zu finden
- Vorgehen geht von **linearer Separierbarkeit** aus
- **Lösung** = Ebene mit maximalem Abstand

Optimierungsproblem mit Nebenbedingungen

$$\arg \min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2 \quad \text{mit} \quad (\mathbf{w}^T \mathbf{x}_i + b) y_i \geq 1$$

Finde  $\mathbf{w}, b$  mit  
maximalem Abstand

Bedingungen, dass alle Punkte  
auf der richtigen Seite liegen

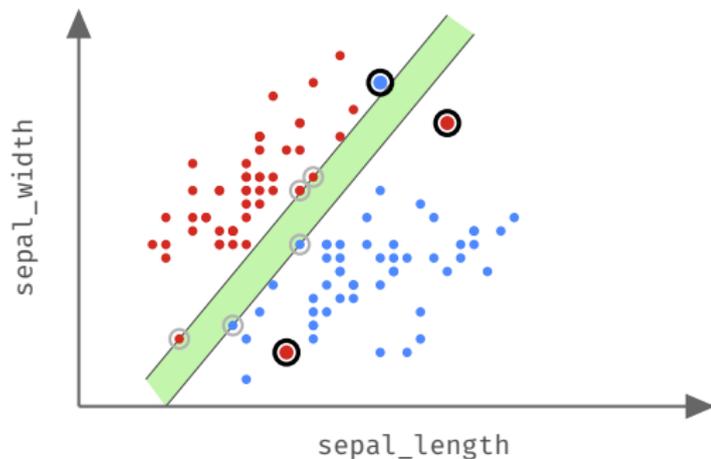
## Optimierung mit Nebenbedingungen

$$\arg \min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2 \quad \text{mit} \quad (\mathbf{w}^T \mathbf{x}_i + b) y_i \geq 1$$

lässt sich mit Lagrange-Optimierung formulieren als

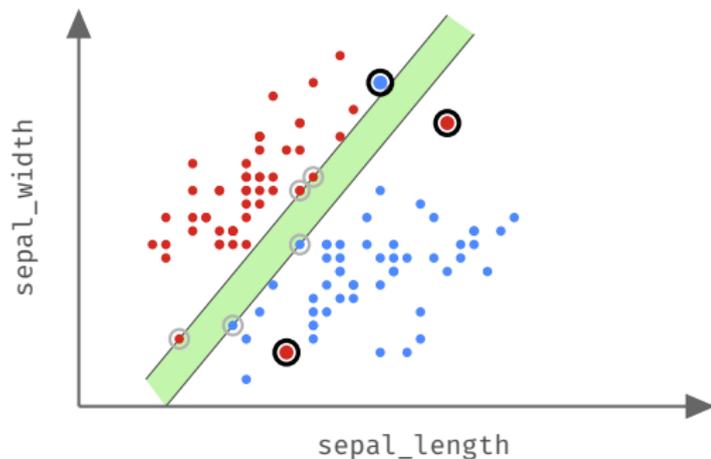
$$L(\mathbf{a}, \mathbf{w}, b) = \frac{1}{2} \|\mathbf{w}\|^2 + \sum_{i=1}^n a_i \left[ y_i (\mathbf{w}^T \mathbf{x}_i + b) - 1 \right]$$

## Lineare SVM geht von **linearer Separierbarkeit** aus



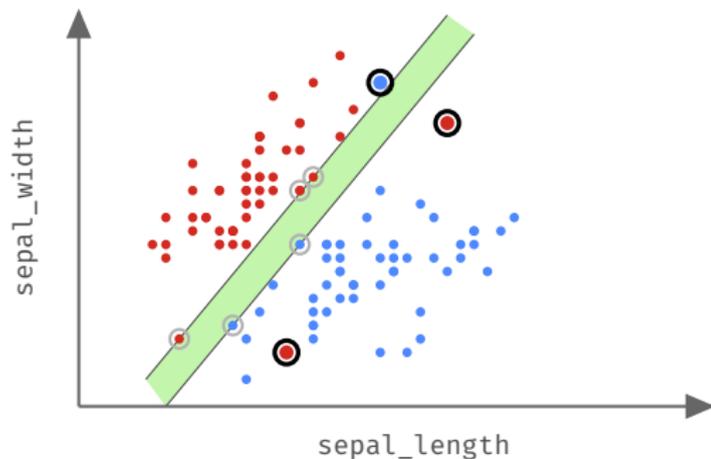
- Was ist, wenn es Ausnahmen in den Daten gibt?

## Lineare SVM geht von **linearer Separierbarkeit** aus



- Was ist, wenn es Ausnahmen in den Daten gibt?
- Dann sind einige Bedingungen verletzt  $\Rightarrow$  keine Lösung

## Lineare SVM geht von **linearer Separierbarkeit** aus

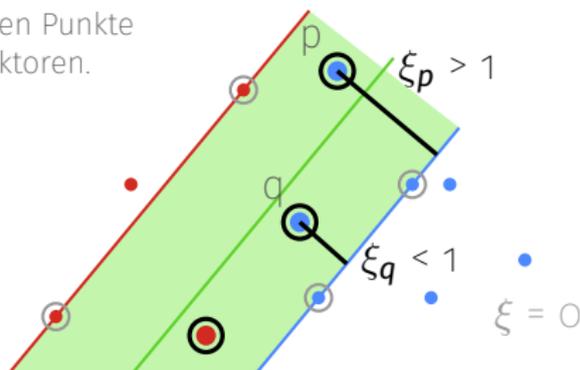


- Was ist, wenn es Ausnahmen in den Daten gibt?
- Dann sind einige Bedingungen verletzt  $\Rightarrow$  keine Lösung

**Wie kommen wir trotzdem zu einer Lösung?**

## Idee: Strafpunkte für Ausnahmen im Trainingsdatensatz

Alle eingekreisten Punkte sind Stützvektoren.



- Punkte im richtigen Bereich bekommen ein  $\xi = 0$  (keine Strafe)
- Punkte auf richtiger Seite im Margin bekommen ein  $\xi < 1$  (leichte Strafe)
- Punkte auf falscher Seite bekommen ein  $\xi > 1$  (große Strafe)

**Idee: Strafpunkte für Ausnahmen im Trainingsdatensatz**

- Erlaubt kleine Fehler im Training
- **Lösung** = Ebene mit minimalen Strafpunkten

Optimierungsproblem mit Strafpunkten:

$$\arg \min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^n \xi_i$$

- Parameter **C** gibt an, wie stark die Fehler gewertet werden

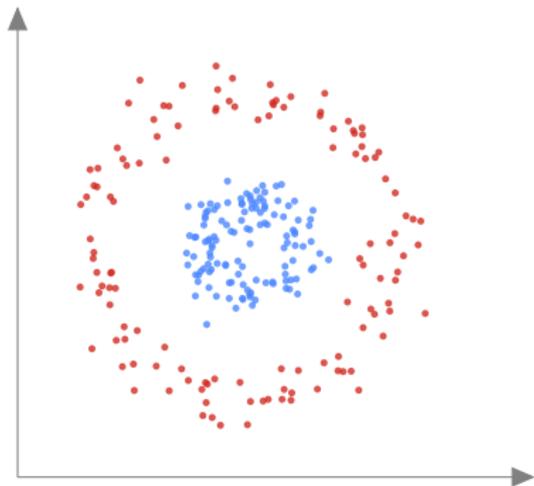
## SVM Modell mit Parameter **C** in SciKit Learn

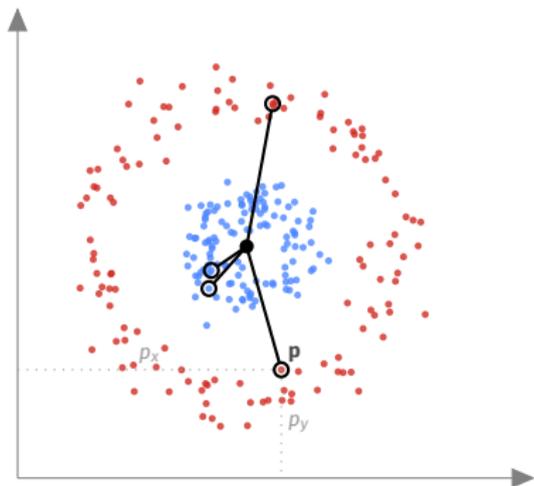
```
# SVC = SupportVectorClassifier
from sklearn.svm import SVC

# Lineare SVM mit C=100
m = SVC(kernel="linear", C=100).
```

- Parameter **C** ist wichtigster *Einstellknopf* für SVMs
- Jede SVM Implementierung enthält Parameter **C**

# Nicht-lineare Separierbarkeit

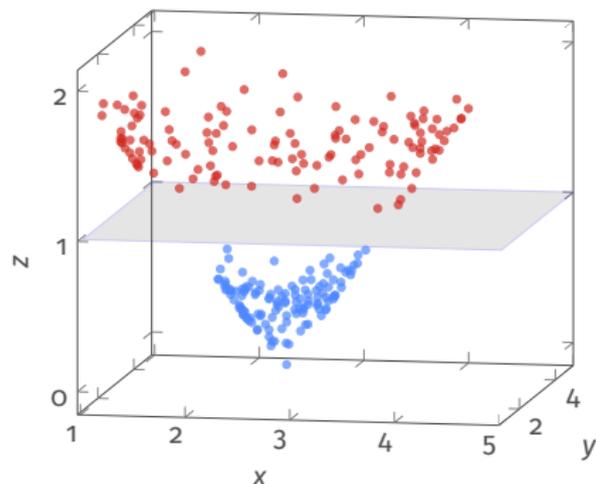
**Problem: Was ist, wenn Strafpunkte nicht ausreichen?**

**Problem: Was ist, wenn Strafpunkte nicht ausreichen?****Mögliche Idee:**

Betrachte die Abstände zum  
Mittelpunkt  $\mathbf{c} = (c_x, c_y)$

$$d(\mathbf{c}, \mathbf{p}) = \sqrt{(c_x - p_x)^2 + (c_y - p_y)^2}$$

## Problem: Was ist, wenn Strafpunkte nicht ausreichen?



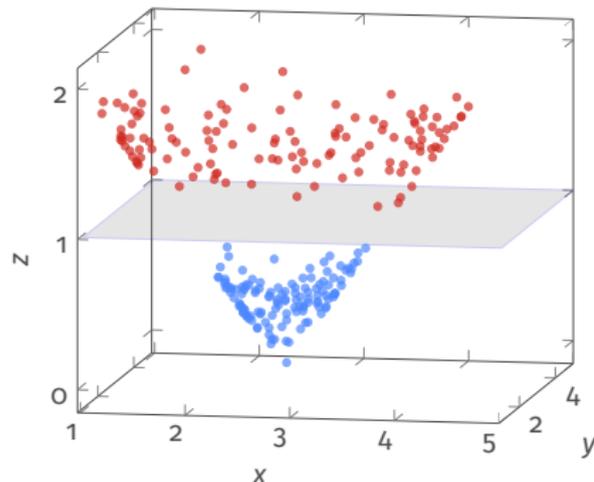
### Mögliche Idee:

Betrachte die Abstände zum  
Mittelpunkt  $\mathbf{c} = (c_x, c_y)$

$$d(\mathbf{c}, \mathbf{p}) = \sqrt{(c_x - p_x)^2 + (c_y - p_y)^2}$$

Und transformiere die Daten in  
einen höherdimensionalen Raum  
mit neuer Dimension  $z$ :

$$\mathbf{p} = \underbrace{(p_x, p_y)}_{\in \mathbb{R}^2} \mapsto \underbrace{(p_x, p_y, d(\mathbf{c}, \mathbf{p}))}_{\in \mathbb{R}^3}$$

**Problem: Was ist, wenn Strafpunkte nicht ausreichen?****Mögliche Idee:**

Betrachte die Abstände zum  
Mittelpunkt  $\mathbf{c} = (c_x, c_y)$

$$d(\mathbf{c}, \mathbf{p}) = \sqrt{(c_x - p_x)^2 + (c_y - p_y)^2}$$

Und transformiere die Daten in  
einen höherdimensionalen Raum  
mit neuer Dimension  $z$ :

$$\mathbf{p} = \underbrace{(p_x, p_y)}_{\in \mathbb{R}^2} \mapsto \underbrace{(p_x, p_y, d(\mathbf{c}, \mathbf{p}))}_{\in \mathbb{R}^3}$$

- Lineare Separierbarkeit in höherdimensionalem Raum
- Transformation in der Theorie meist mit  $\Phi$  bezeichnet

**Problem: Transformation von Daten ist **aufwendig!****

**Idee:** SVM braucht höhere Dimension nur im Skalarprodukt!

$$L(\mathbf{a}, \mathbf{w}, b) = \frac{1}{2} \|\mathbf{w}\|^2 + \sum_{i=1}^n a_i \left[ y_i (\underbrace{\langle \mathbf{w}, \mathbf{x}_i \rangle}_{=: k(\mathbf{w}, \mathbf{x}_i)} + b) - 1 \right]$$

Statt  $\langle \mathbf{w}, \mathbf{x}_i \rangle$  nehmen wir nun  $\langle \Phi(\mathbf{w}), \Phi(\mathbf{x}_i) \rangle$ , d.h. wir berechnen das Skalarprodukt von  $\mathbf{w}$  und  $\mathbf{x}_i$  im höherdimensionalen Raum.

$k(\mathbf{w}, \mathbf{x}_i) = \langle \Phi(\mathbf{w}), \Phi(\mathbf{x}_i) \rangle$  wird als **Kern-Funktion** (*kernel*) bezeichnet.

## Idee: Kern-Funktionen ersparen die Transformation

- Transformation lässt sich in Kern-Funktion integrieren
- Kern-Funktion  $k$  berechnet Skalarprodukt in höher-dimensionalen Räumen
- Mit  $k(\mathbf{v}, \mathbf{w}) = \langle \mathbf{v}, \mathbf{w} \rangle$  ergibt sich die lineare SVM

## Weitere bekannte Kern-Funktionen

- RBF Kern (Radial Basis Functions):

$$k(\mathbf{v}, \mathbf{w}) = \exp(-\gamma(\|\mathbf{v} - \mathbf{w}\|^2)) \text{ , mit freiem Parameter } \gamma$$

- Polynomielle Kern-Funktion

$$k(\mathbf{v}, \mathbf{w}) = (\langle \mathbf{v}, \mathbf{w} \rangle + r)^p \text{ , für } r \geq 0, p \geq 1$$

## Kern-Funktionen natürlich auch in SciKit-Learn

```
from sklearn.svm import SVC

rbf1 = SVC(gamma=2)    # default: RBF Kern

rbf2 = SVC(kernel="rbf", gamma=2)

# polynomieller kern mit Parametern r und p:
poly = SVC(kernel="polynomial", degree=p, coef0=r)

# lineare SVM
lin = SVC(kernel="linear")
```

## Kern-Funktionen natürlich auch in SciKit-Learn

```
from sklearn.svm import SVC

rbf1 = SVC(gamma=2)    # default: RBF Kern

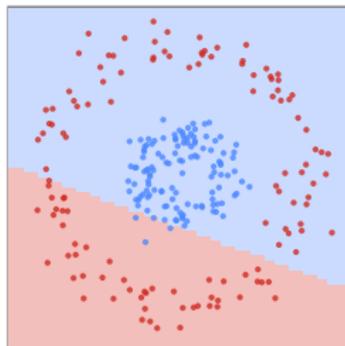
rbf2 = SVC(kernel="rbf", gamma=2)

# polynomieller kern mit Parametern r und p:
poly = SVC(kernel="polynomial", degree=p, coef0=r)

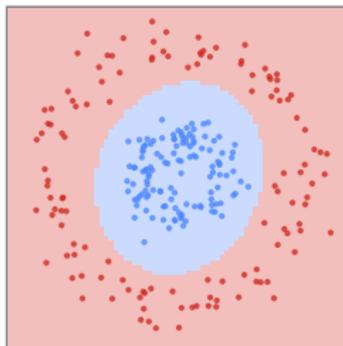
# lineare SVM
lin = SVC(kernel="linear")
```

**Dazu kommt noch Parameter C für die Straf-Kosten!**

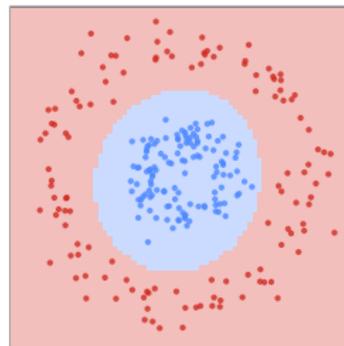
## Beispiele: SVM Model mit verschiedenen Parametern



Linearer Kern,  $C=100$



Polynomieller Kern,  
 $\text{degree}=3$ ,  $\text{coefo}=2$ ,  $C=100$



RBF Kern,  $\text{gamma}=0.1$ ,  $C=100$

## Beispiele: SVM Model mit verschiedenen Parametern

- Die vorherigen Plots wurden auf synthetischen Daten erstellt
- Hilfreich, um selbst zu experimentieren
- Versuchen Sie die Beispiele im Notebook nachzuvollziehen!



Probieren Sie es im Notebook aus!

Notebook: [Vorlesung/V8-Klassifikation-SVM-ZweiKreise](#)