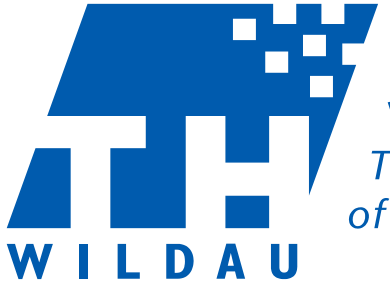


Technische
Hochschule
Wildau [FH]
Technical University
of Applied Sciences

Maschinelles Lernen

Komplexere Modellfamilien

Prof. Dr. Rainer Stollhoff

- Wiederholung
  - Lineare Regression
  - Klassifikationsbäume / Ensemlemethoden
- Allgemein
  - Regression - Klassifikation
  - Exkurs: Parametrisch / Nicht-Parametrisch
- Generalisierte Lineare Modelle
- SVM
- Neuronale Netzwerke

Multivariate Lineare Regression – Gradientenabstieg

Technische Hochschule Wildau
Technical University of Applied Sciences
WILDAU

Aufgabe: Regression, d.h. Vorhersage $\hat{y} = \hat{y}(x) = f(x)$

Erfahrung: Datensatz $(x_i, y_i)_{i=1}^n$ mit $x_i = (x_{i,1}, x_{i,2}, \cdots, \boldsymbol{x_{i,n}})$

Qualität: Verlustfunktion: $L(y, \hat{y}) = \sum_{i=1}^n (y_i - \hat{y_i})^2 = \sum_{i=1}^n (y_i - f(x_i; \theta))^2 = L(\theta)$

Maschine: Regression mit $f(\boldsymbol{x}; \boldsymbol{\theta}) = \theta_0 + \theta_1 \cdot x_1 + \theta_2 \cdot x_2 + \cdots + \boldsymbol{\theta_m} \cdot \boldsymbol{x_m}$

Lernen: Finde Werte für $\theta = (\theta_0, \theta_1, \theta_2, \cdots, \boldsymbol{\theta_m})$, die die quadratische Verlustfunktion minimieren

Durch geeignete Wahl von θ in einem iterativen Prozess (Gradientenabstiegsverfahren):

1. Wähle Startwert z.B. $\theta^0 = \begin{pmatrix} 0 \\ 1 \\ \vdots \\ \boldsymbol{1} \end{pmatrix}$

2. Berechne Gradienten

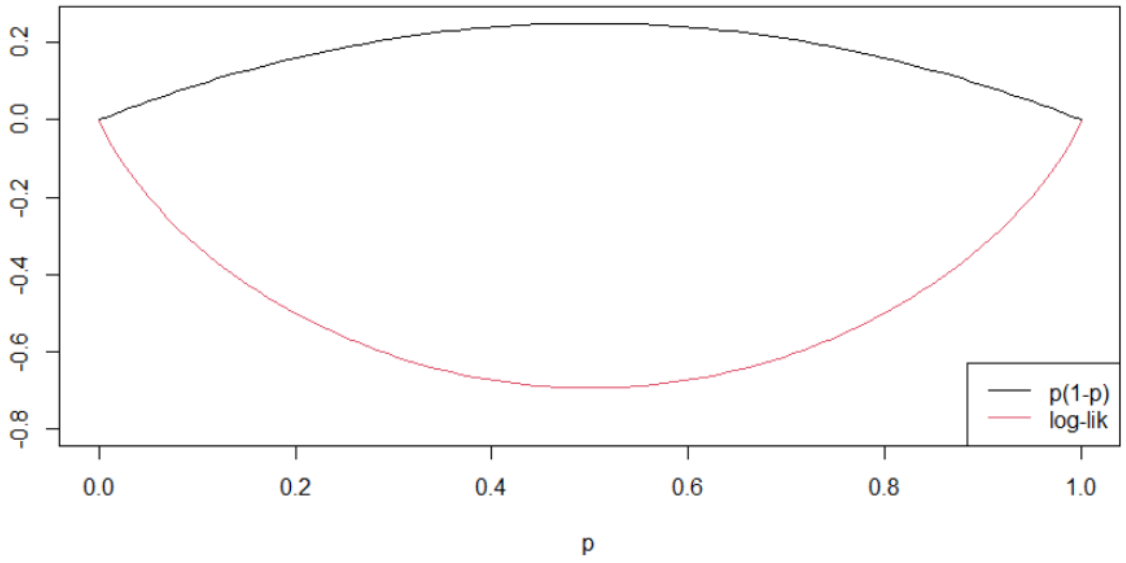
$$\nabla L(\theta^0) = \begin{pmatrix} \frac{\partial}{\partial \theta_0} L(\theta^0) \\ \frac{\partial}{\partial \theta_1} L(\theta^0) \\ \vdots \\ \boldsymbol{\frac{\partial}{\partial \theta_m} L(\theta^0)} \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^n (y_i - (\theta_0 + \theta_1 \cdot x_1 + \theta_2 \cdot x_2 + \cdots + \boldsymbol{\theta_m} \cdot \boldsymbol{x_m})) \cdot (-2 \cdot 1) \\ \sum_{i=1}^n (y_i - (\theta_0 + \theta_1 \cdot x_1 + \theta_2 \cdot x_2 + \cdots + \boldsymbol{\theta_m} \cdot \boldsymbol{x_m})) \cdot (-2 \cdot x_{i,1}) \\ \vdots \\ \boldsymbol{\sum_{i=1}^n (y_i - (\theta_0 + \theta_1 \cdot x_1 + \theta_2 \cdot x_2 + \cdots + \theta_m \cdot x_m)) \cdot (-2 \cdot x_{i,m})} \end{pmatrix}$$

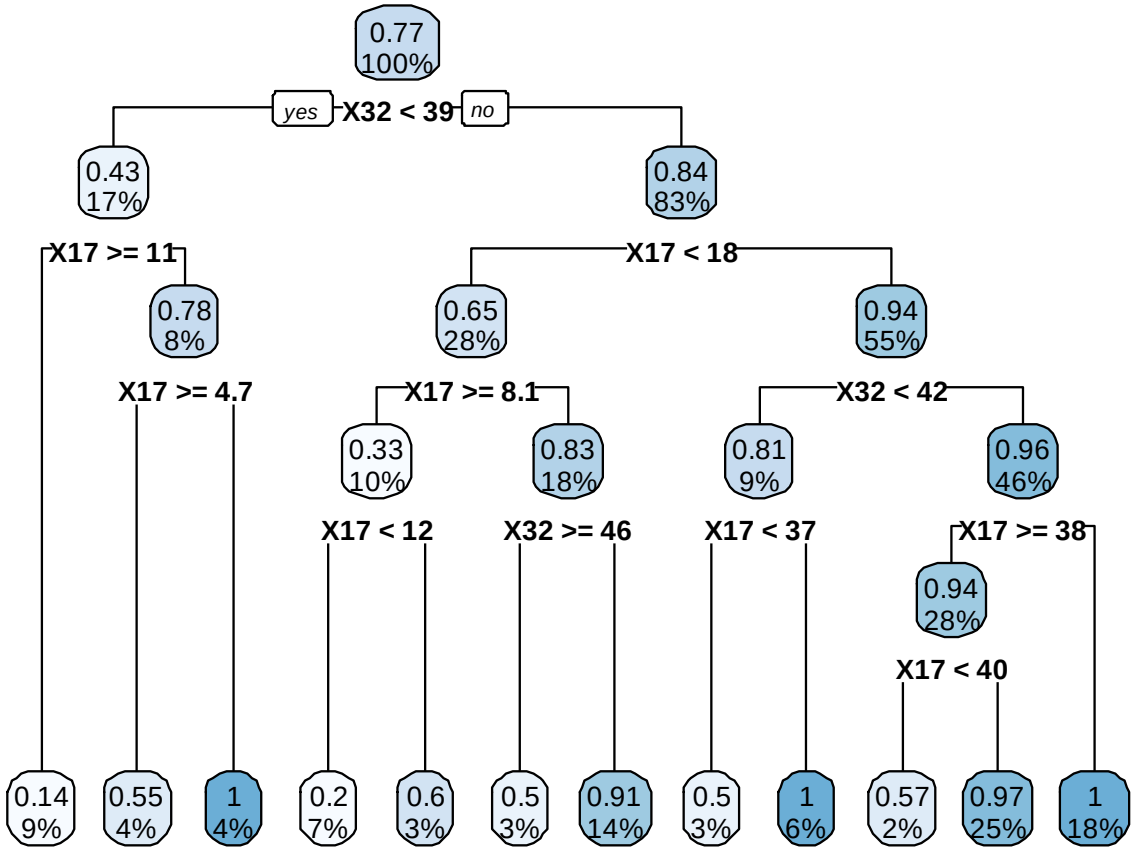
3. Update $\theta^{t+1} = \theta^t + \alpha \cdot \nabla L(\theta^t)$

Klassifikationsbäume

- Idee: Rekursive Partitionierung / Wiederholtes Aufteilen
- Teilungsregel:
  In jedem neuen Knoten möglichst die gleiche Klasse,
  d.h. mit $p_K = {}^{\#\{\text{i in K: } y_i=1\}}/_{\#\{i \; in \; K\}}$
  - Minimiere Brier score $p_K \cdot (1 - p_K)$
  - Maximiere log-lik

$$\sum_{i \; in \; K: y_i=1} \log p_K + \sum_{i \; in \; K: y_i=0} \log(1 - p_K)$$





Ensemble Methoden

Technische Hochschule Wildau
Technical University of Applied Sciences
WILDAU

- Idee: „Swarm Intelligence“ oder „Wisdom of the Crowd“
  - Erzeuge wiederholt einfache Modelle
  - Aggregiere die einzelnen Vorhersagen z.B. mit Mittelwertbildung oder Mehrheitsentscheid
- Bagging = Bootstrap-Aggregating
  - Erzeuge Modelle auf Bootstrap Stichproben
- RandomForest
  - Erzeuge Bäume mit zufälligen Parametern (hier: Wahl der Partitionierung in einem Knoten) auf Bootstrap Stichproben
- Boosting
  - Erzeuge Modelle auf iterativ gewichteten Datensätzen
  - Erhöhe die Gewichte von Beobachtungen, die falsch vorhergesagt wurden

- Wiederholung
  - Lineare Regression
  - Klassifikationsbäume / Ensemlemethoden
- **Allgemein**
  - Regression - Klassifikation
  - Exkurs: Parametrisch / Nicht-Parametrisch
- Generalisierte Lineare Modelle
- SVM
- Neuronale Netzwerke

Ausblick: Multivariate Regression – Gradientenabstieg

Technische Hochschule Wildau
Technical University of Applied Sciences
WILDAU

Aufgabe: Regression, d.h. Vorhersage $\hat{y} = \hat{y}(x) = f(x)$

Erfahrung: Datensatz $(x_i, y_i)_{i=1}^{n}$ mit $x_i = (x_{i,1}, x_{i,2}, \cdots, x_{i,n})$

Qualität: Verlustfunktion: $\boldsymbol{L(y, \hat{y}) = L(\theta)}$

Maschine: Regression mit $\mathbf{f(x; \theta)}$

Lernen: Finde Werte für $\theta = (\theta_0, \theta_1, \theta_2, \cdots, \theta_m)$, die die quadratische Verlustfunktion minimieren

Durch geeignete Wahl von θ in einem iterativen Prozess (Gradientenabstiegsverfahren):

1. Wähle Startwert z.B. $\theta^0 = \begin{pmatrix} 0 \\ 1 \\ \vdots \\ 1 \end{pmatrix}$

2. Berechne Gradienten

$$\nabla L(\theta^0) = \begin{pmatrix} \frac{\partial}{\partial \theta_0} L(\theta^0) \\ \frac{\partial}{\partial \theta_1} L(\theta^0) \\ \vdots \\ \frac{\partial}{\partial \theta_m} L(\theta^0) \end{pmatrix} = \begin{pmatrix} \boldsymbol{\frac{d}{df} L(f(\theta^0)) \cdot \frac{\partial}{\partial \theta_0} f(x; \theta^0)} \\ \boldsymbol{\frac{d}{df} L(f(\theta^0)) \cdot \frac{\partial}{\partial \theta_1} f(x; \theta^0))} \\ \vdots \\ \boldsymbol{\frac{d}{df} L(f(\theta^0)) \cdot \frac{\partial}{\partial \theta_m} f(x; \theta^0))} \end{pmatrix}$$

3. Update $\theta^{t+1} = \theta^t + \boldsymbol{A} \cdot \nabla L(\theta^t)$

Vorhersage anhand von Wahrscheinlichkeiten

- Fehlklassifikationsrate

$$\text{err} = \frac{1}{n}\sum_i 1_{(y_i \neq \hat{y}_i)}$$

- Quadrat. Fehler der Wahrs.

$$L(p, \hat{p}) = \frac{1}{n}\sum_i (p_i - \hat{p}_i)^2$$

- Brier-Score

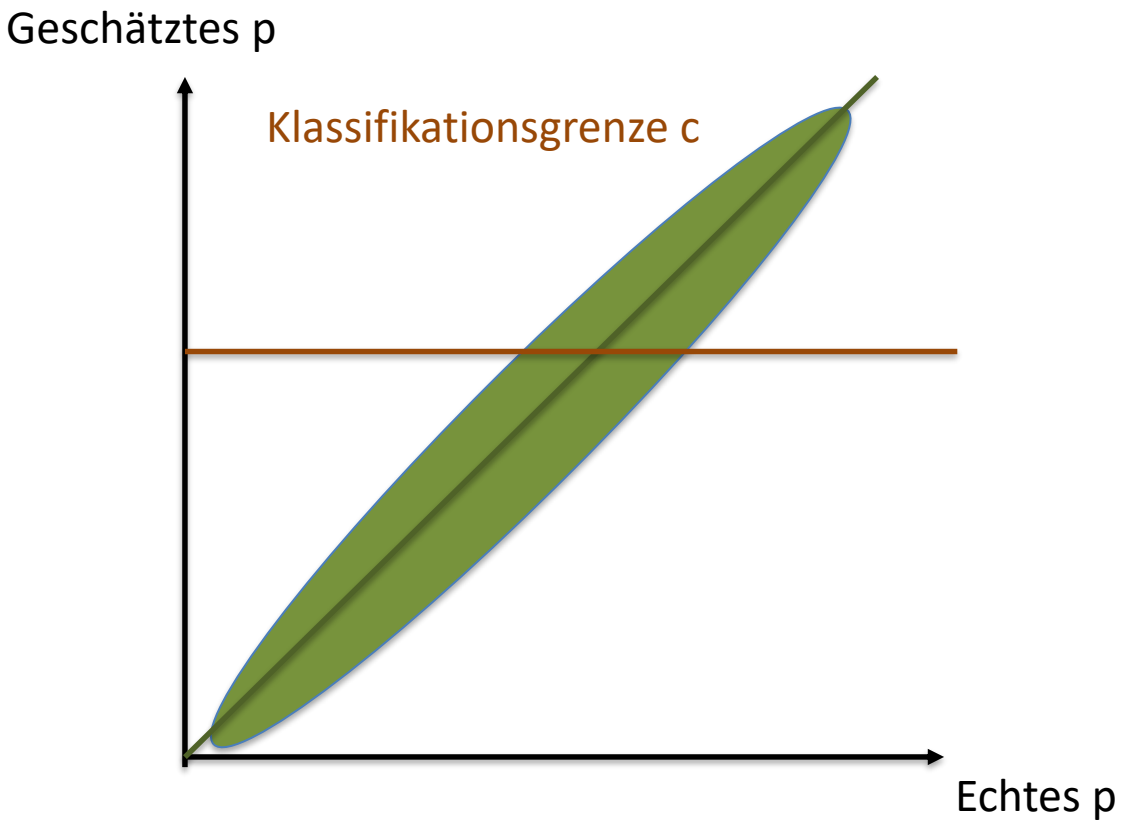
$$L(y, \hat{p}) = \frac{1}{n}\sum_i (y_i - \hat{p}_i)^2$$

- Likelihood

$$L(y, \hat{p}) = \prod_{i:y_i=1} \hat{p}(x) \prod_{i:y_i=0} (1 - \hat{p}(x))$$

- Log-Likelihood

$$\mathrm{l}(y, \hat{p}) = \sum_{i:y_i=1} \log \hat{p}(x_i) + \sum_{i:y_i=0} \log(1 - \hat{p}(x_i))$$



Klassifikation mittels Linearer Regression?

Technische Hochschule Wildau
Technical University of Applied Sciences
WILDAU

- Modellfamilie

$$f(x;\theta) = \theta_0 + \sum_{j=1}^{k} x_j\theta_j = x \cdot \theta$$

  - Für gewöhnlich werden Parameter mit β statt mit θ bezeichnet
- Mit Klassifikation

$$\hat{y}(x) = \begin{cases} 1 & \text{falls } f(x;\theta) \geq 0{,}5 \\ 0 & \text{falls } f(x;\theta) < 0{,}5 \end{cases}$$

- Minimiere die Verlustfunktion

$$L(y,\hat{y}) = (y - \hat{y})^2$$
$$L(y,f) = (y - x \cdot \theta)^2$$

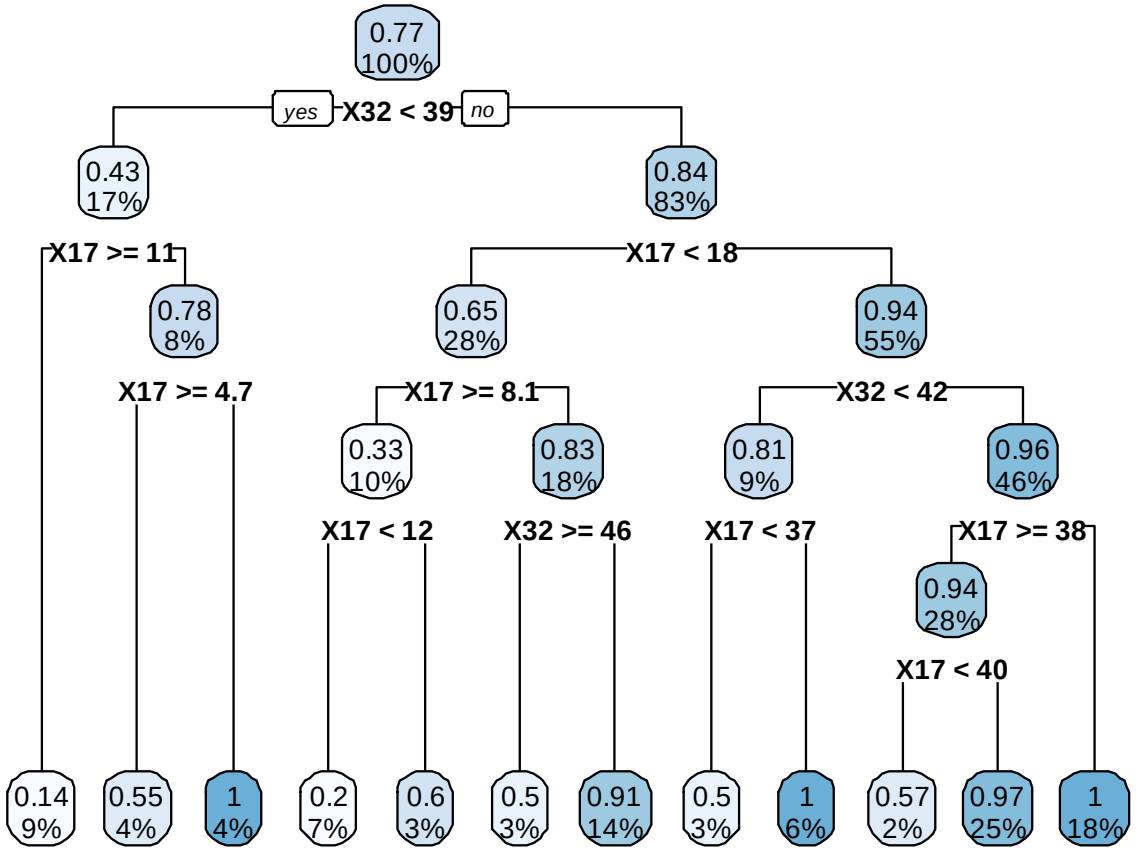
für den Datensatz $(x_i, y_i)_{i=1}^n$

$$L(y,\hat{y}) = \sum_{i=1}^{n} (y_i - \hat{y_i})^2$$
$$\neq \sum_{i=1}^{n} (y_i - x_i \cdot \theta)^2 = L(\theta)$$

Regression mit Bäumen?

- Idee: Rekursive Partitionierung / Wiederholtes Aufteilen
- Teilungsregel:
  Neue Knoten mit möglichst geringer Varianz,
  d.h. mit $\bar{y}_{K_1} = {}^{1}/_{\#K_1} \sum_{i\ in\ K_1} y_i$ und analog $\bar{y}_{K_2}$
  - Minimiere quadratischen Fehler

$$\sum_{i\ in\ K_1} \left(y_i - \bar{y}_{K_1}\right)^2 + \sum_{i\ in\ K_2} \left(y_i - \bar{y}_{K_2}\right)^2$$



Exkurs: Klassifikation – Maximum-Likelihood-Schätzer mit Gradientenabstieg

Technische Hochschule Wildau
Technical University of Applied Sciences

Aufgabe: Klassifikation, d.h. Vorhersage $\hat{y} = \hat{y}(x) \in \{0,1\}$
mittels Vorhersage der bedingten Wahrscheinlichkeit $\mathbf{f}(\boldsymbol{x};\boldsymbol{\theta}) = \hat{p}(y = 1|x;\theta)$ und Grenzwert

Erfahrung: Datensatz $(x_i, y_i)_{i=1}^n$

Qualität: (Likelihood) oder log-likelihood

$$L(\theta) = \sum_{i:y_i=1} \log \hat{p}(x_i;\theta) + \sum_{i:y_i=0} \log(1 - \hat{p}(x_i;\theta))$$

Lernen: Finde einen **Wert für $\boldsymbol{\theta}$** , der die log-likelihood minimiert (bzw. likelihood maximiert)
Durch geeignete **Wahl von $\boldsymbol{\theta}$** in einem iterativen Prozess (Gradientenabstiegsverfahren):

1. Wähle Startwert z.B. $\theta^0 = \begin{pmatrix} 0 \\ 1 \\ \vdots \\ \mathbf{1} \end{pmatrix}$

2. Berechne Gradienten $\nabla L(\theta^0) = \begin{pmatrix} \frac{\partial}{\partial \theta_0} L(\theta^0) \\ \frac{\partial}{\partial \theta_1} L(\theta^0) \\ \vdots \\ \boldsymbol{\frac{\partial}{\partial \theta_n} L(\theta^0)} \end{pmatrix} = \begin{pmatrix} \frac{d}{df} L(f(\theta^0)) \cdot \frac{\partial}{\partial \theta_0} f(x;\theta^0) \\ \frac{d}{df} L(f(\theta^0)) \cdot \frac{\partial}{\partial \theta_1} f(x;\theta^0)) \\ \vdots \\ \frac{d}{df} L(f(\theta^0)) \cdot \frac{\partial}{\partial \theta_n} f(x;\theta^0)) \end{pmatrix}$

3. Update $\theta^{t+1} = \theta^t + A \cdot \nabla L(\theta^t)$

Vergleich: Parametrisch und Nicht-Parametrisch

Technische Hochschule Wildau
Technical University of Applied Sciences
WILDAU

	Parametrisch	Nicht-Parametrisch
Art und Anzahl Parameter	vorab festgelegt	wird iterativ bestimmt
Beispiele	Lineare Regression Logistische Regression Neuronale Netzwerke	Klassifikationsbäume Regressionsbäume Ensemble-Verfahren Support Vector Machines
Optimierung	analytisch oder iterativ	iterativ
Interpretierbarkeit	direkt über den Wert der Parameter	indirekt über Auswertungsfunktionen
Komplexität	vorab festgelegt	variabel
Flexibilität	steigt mit Komplexität	variabel

- Wiederholung
  - Lineare Regression
  - Klassifikationsbäume / Ensemlemethoden
- Allgemein
  - Regression - Klassifikation
  - Exkurs: Parametrisch / Nicht-Parametrisch
- **Generalisierte Lineare Modelle**
- SVM
- Neuronale Netzwerke

Logistische Regression

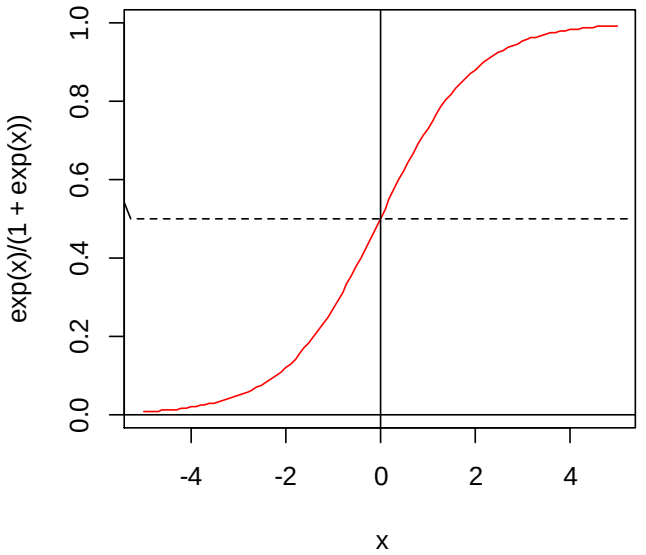
- Linearer Kern

$$f(x;\theta) = \theta_0 + \sum_{j=1}^{k} x_j\theta_j = x \cdot \theta$$

- Logistische Transformation

$$\hat{p}(y = 1|x;\theta) = \hat{p}(x;\theta) = \frac{\exp(f(x;\theta))}{1+\exp(f(x;\theta))}$$

$$\frac{\log(\hat{p}(x;\theta))}{\log(1-\hat{p}(x;\theta))} = f(x;\theta)$$



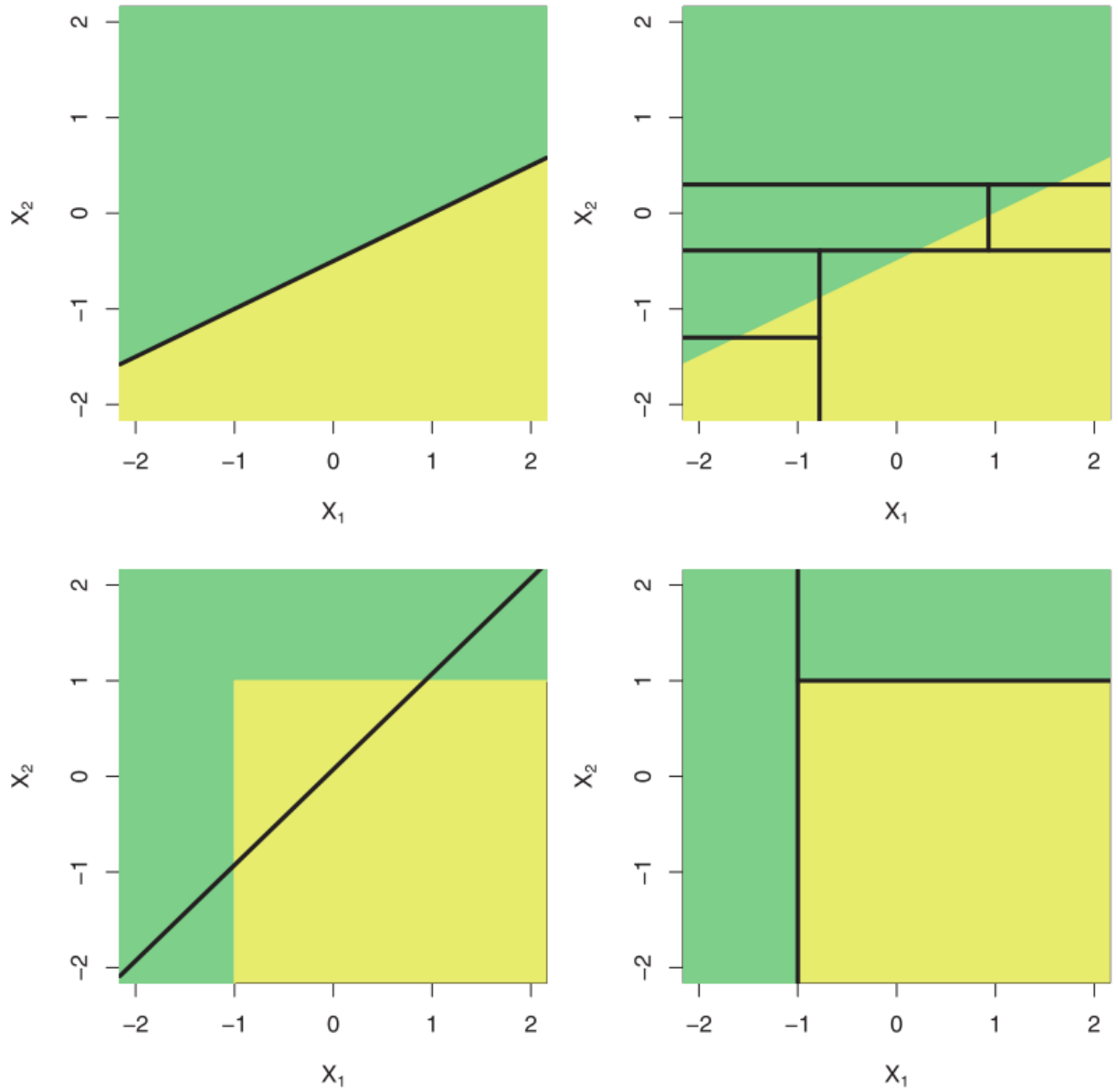
- Wähle die Parameter θ so, dass Sie die Log-Likelihood

für den Datensatz $(x_i, y_i)_{i=1}^{n}$ maximieren

$$\log\big(L(\theta)\big)$$

$$= \sum_{i:y_i=1} \log \hat{p}(x_i;\theta) + \sum_{i:y_i=0} \log(1-\hat{p}(x_i;\theta))$$

Vergleich Logistische Regression / Baum



James et al., 2013

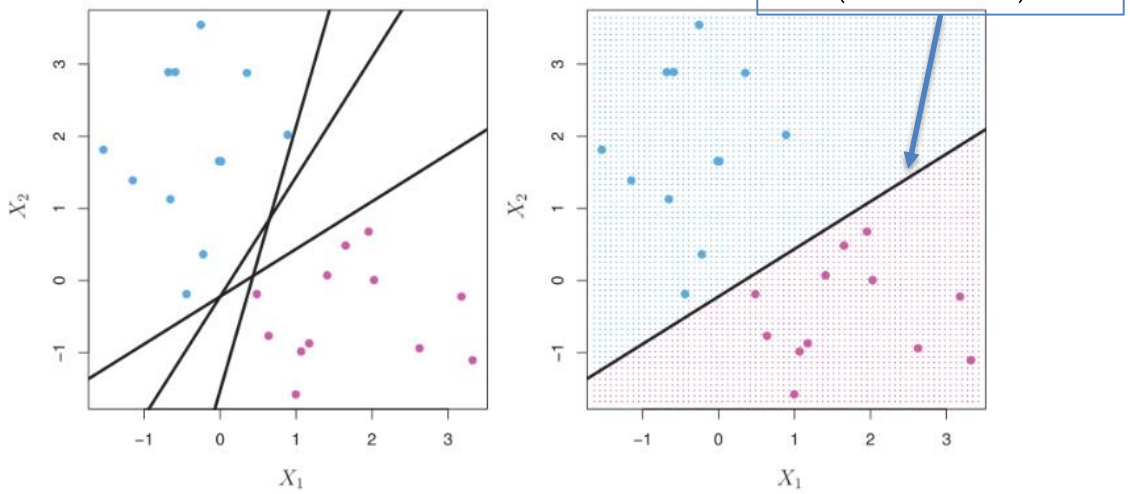
- Wiederholung
  - Lineare Regression
  - Klassifikationsbäume / Ensemlemethoden
- Allgemein
  - Regression - Klassifikation
  - Exkurs: Parametrisch / Nicht-Parametrisch
- Generalisierte Lineare Modelle
- **Support Vector Machines (SVM)**
- Neuronale Netzwerke

Support Vector Machines - Hyperebene

- Grundidee: Trennende Hyperebenen

$$y \in \{-1,1\}$$

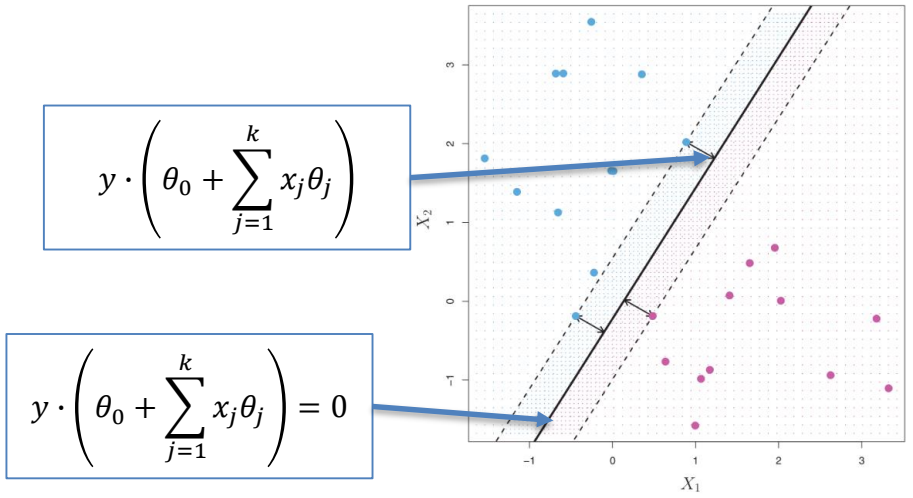
$$y \cdot \left(\theta_0 + \sum_{j=1}^{k} x_j \theta_j \right) \geq 0$$



- Erweiterung: Abstand zur Grenze (M: margin) maximieren

$$y \cdot \left(\theta_0 + \sum_{j=1}^{k} x_j \theta_j \right) \geq M$$

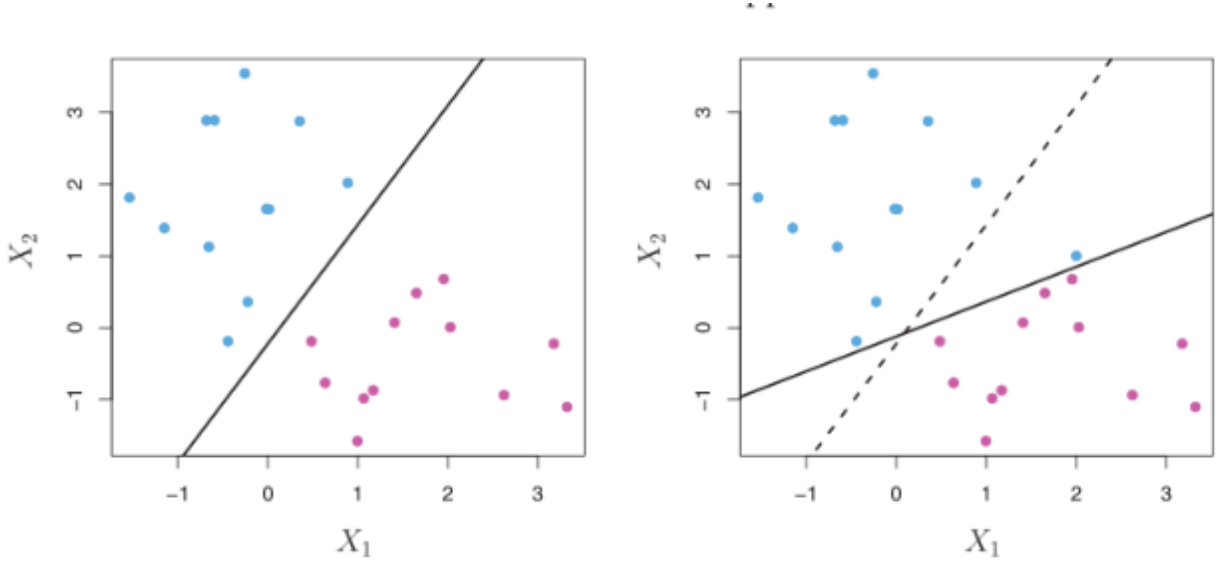
Mit Nebenbedingung $\left(\sum_{j=0}^{k} {\theta_j}^2 \right) = 1$



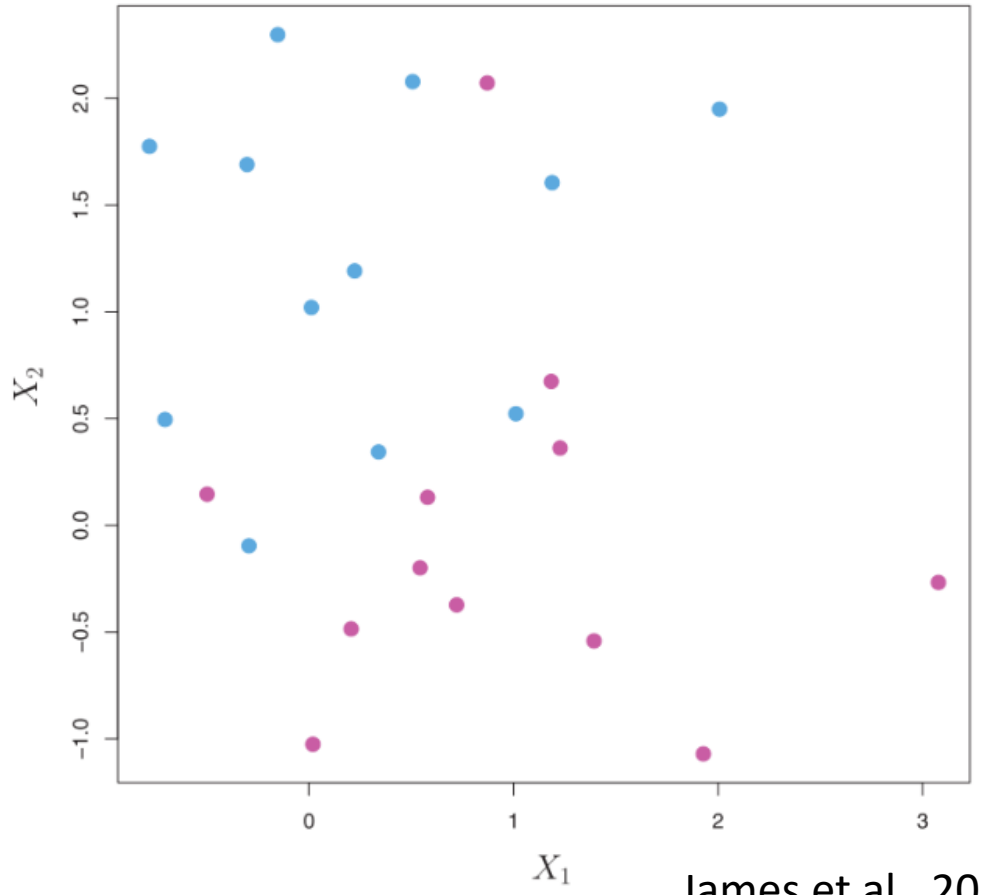
James et al., 2013

Suppor Vector Machines - Hyperebene

- Problemfall 1: Robustheit gegenüber Ausreißern



- Problemfall 2: Nicht-separierbare, überlappende Klassen



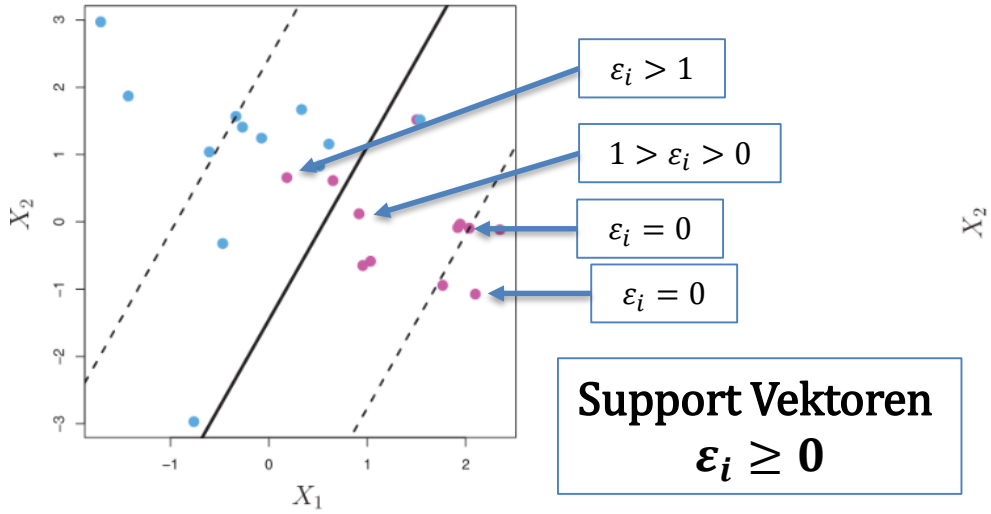
James et al., 2013

Support Vector Machines - SVC

- Grundidee Support Vector Classifier: Schlupfvariablen ε_i ermöglichen trennende Hyperebenen, Margin-Maximierung

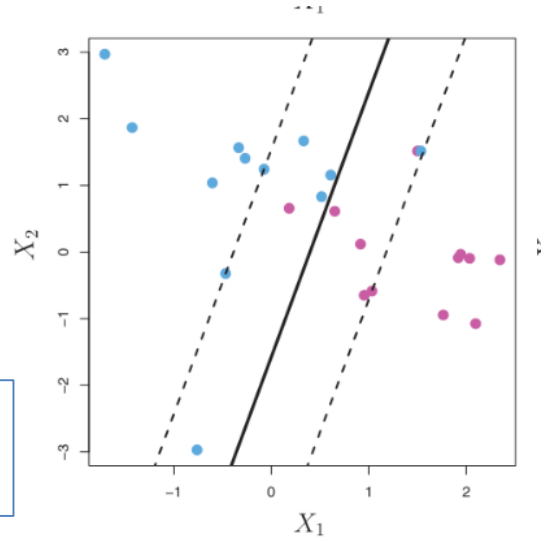
$$y_i \cdot \left(\theta_0 + \sum_{j=1}^{k} x_{i,j}\theta_j \right) \geq M(1 - \varepsilon_i)$$

Mit Nebenbedingungen $\left(\sum_{j=0}^{k} {\theta_j}^2\right) = 1$, $\varepsilon_i \geq 0$ und $\left(\sum_{i=1}^{n} \varepsilon_i\right) \leq C$



Support Vektoren
$\varepsilon_i \geq 0$

Großes C = tolerant bei Fehlern
= großes M

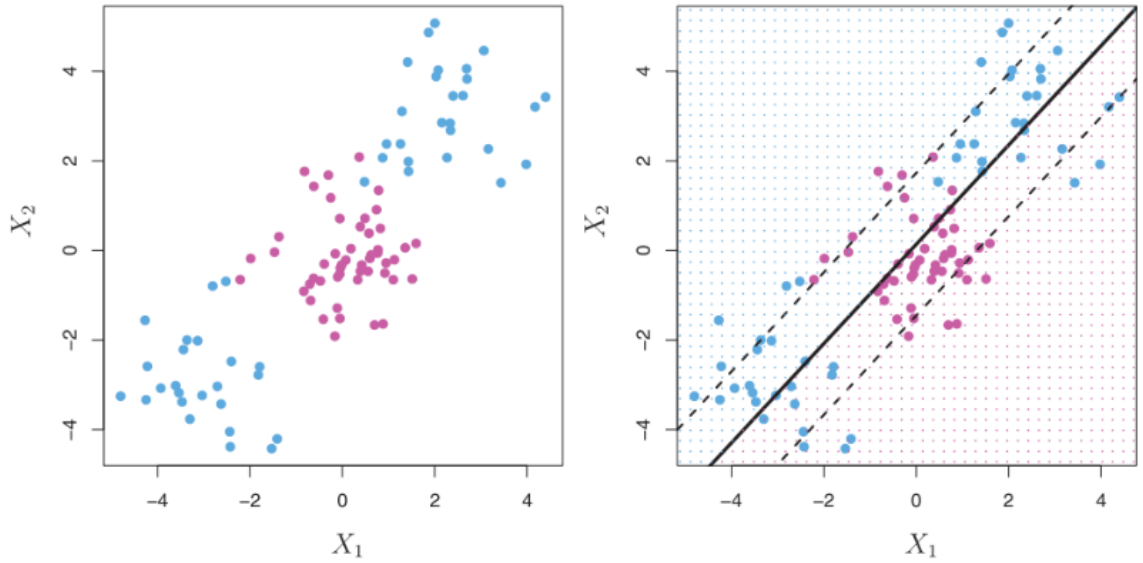


Kleines C = wenig Fehler
= kleines M

James et al., 2013

Support Vector Machines - SVC

- Problemfall: Lineare Trennung nicht möglich



- Lösungsansatz: Variablentransformation

$$y_i \cdot \left(\theta_0 + \sum_{j=1}^{k} x_{i,j}\theta_{j,1} + \sum_{j=1}^{k} {x_{i,j}}^2 \theta_{j,2} \right) \geq M(1 - \varepsilon_i)$$

Mit Nebenbedingungen $\left(\sum_{j=0}^{k} {\theta_{j,1}}^2 + \sum_{j=0}^{k} {\theta_{j,2}}^2 \right) = 1 \quad , \varepsilon_i \geq 0 \quad$ und $\quad (\sum_{i=1}^{n} \varepsilon_i) \leq C$

James et al., 2013

Support Vector Machines

Technische Hochschule Wildau
Technical University of Applied Sciences
WILDAU

Der Kernel-Trick:

- verwende Skalarprodukt mit Beobachtungen $(x_i)_{i=1}^n$

$$\langle x, x_i \rangle = \sum_{j=1}^{k} x_j x_{i,j}$$

- als Ersatz für Parameter im linearen Modell

$$\theta_0 + \sum_{j=1}^{k} x_j \theta_j = \theta_0 + \sum_{i=1}^{n} \sum_{j=1}^{k} \alpha_i x_j x_{i,j} = \theta_0 + \sum_{i=1}^{n} \alpha_i \langle x, x_i \rangle$$

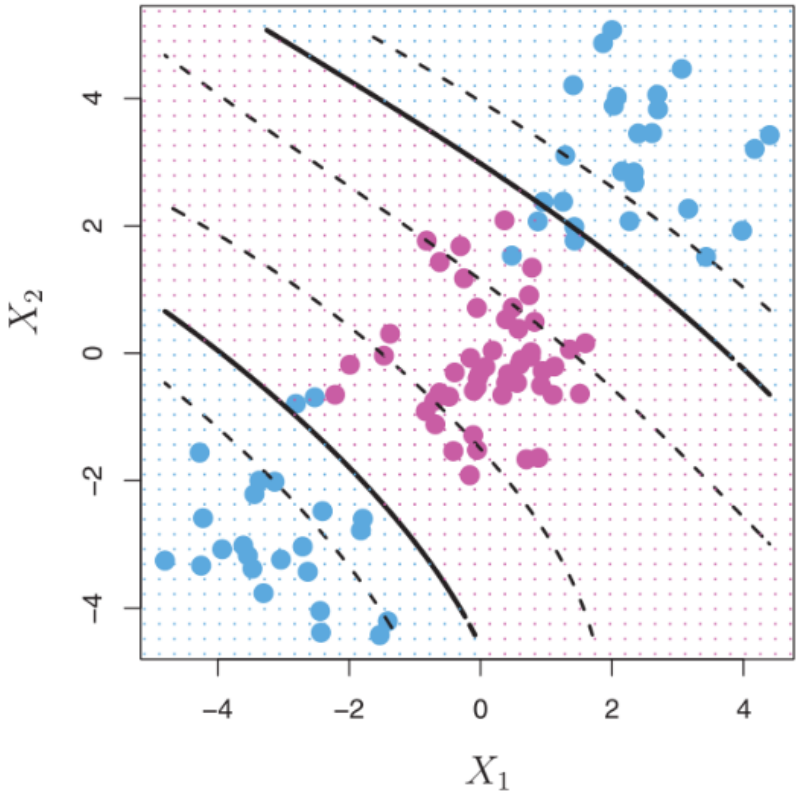
wobei $\alpha_i \neq 0$ nur für Support-Vektoren und $\theta_j = \sum_{i=1}^{n} \alpha_i x_{i,j}$

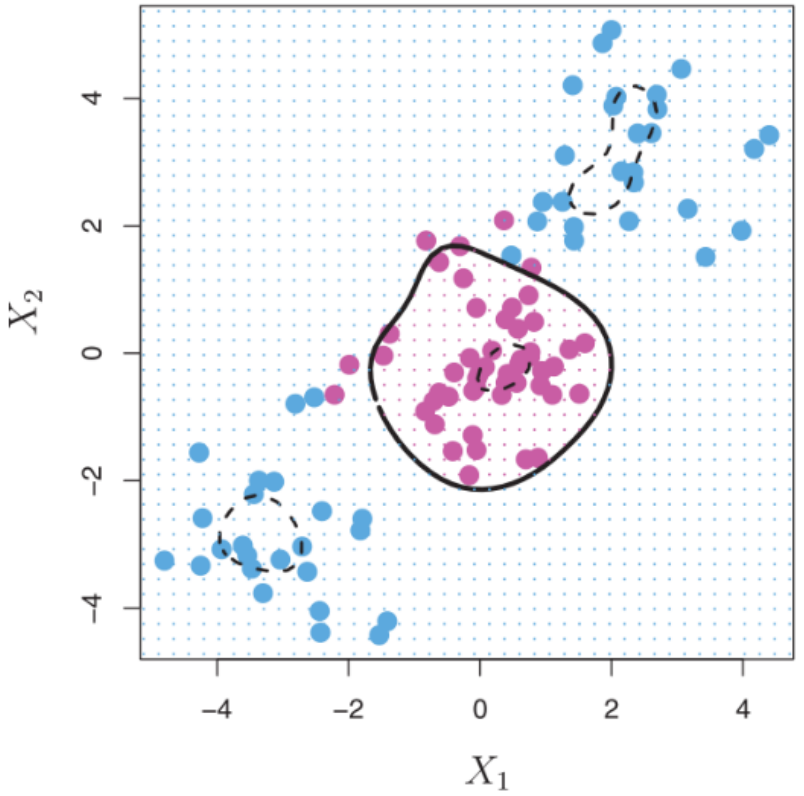
- Und ersetze Skalarprodukt $\langle x, x_i \rangle$ durch Kernel $K(x, x_i)$
  - Linearer Kern $K(x, x_i) = \langle x, x_i \rangle$
  - Polynomieller Kern $K(x, x_i) = (c + \gamma \langle x, x_i \rangle)^d$
  - Radiale Basis Kern $K(x, x_i) = exp\left(-\gamma \cdot \sum_{j=1}^{k} \langle x - x_i, x - x_i \rangle \right)$
- Damit: Nicht-lineare Trennung mit linearer Darstellung

$$\theta_0 + \sum_{i=1}^{n} \alpha_i K(x, x_i)$$

Support Vector Machines

Technische Hochschule Wildau
Technical University of Applied Sciences
ILDAU





$$K(x, x_i) = (c + \gamma\langle x, x_i\rangle)^3$$

$$K(x, x_i) = exp\left(-\gamma \cdot \sum_{j=1}^{k}(x_j - x_{i,j})^2\right)$$

Vergleich logReg und SVM

Technische Hochschule Wildau
Technical University of Applied Sciences
WILDAU

logReg

- Linearer Kern

$$f(x;\theta) = \theta_0 + \sum_{j=1}^{k} x_j\theta_j = x \cdot \theta$$

- Parameter entsprechen Einfluss der Variablen
- Nichtlineares y durch logistische Transformation

$$\hat{p}(y = 1|x) = \frac{\exp(f(x;\theta))}{1+\exp(f(x;\theta))}$$

- Nichtlineare Trennung durch Interaktionen und Transformationen der Variablen (z.B. MFP)

```
+ X5 * X6 + I(X7^2)
```

SVM

$$y_i \cdot \left(\theta_0 + \sum_{i=1}^{n} \alpha_i \cdot K\,(x, x_i)\right) \geq M(1 - \varepsilon_i)$$

- Trennende Hyperebene
- Maximierung der Margin
- Schlupfvariablen ε_i
- Support-Vektoren x_i mit $\varepsilon_i \geq 0$ und $\alpha_i \geq 0$
- Parameter entsprechen Einfluss der Ähnlichkeit zu Support Vektoren
- Nicht-lineare Kernelfunktionen

Technische Hochschule Wildau
Technical University of Applied Sciences
WILDAU

- Wiederholung
  - Lineare Regression
  - Klassifikationsbäume / Ensemlemethoden
- Allgemein
  - Regression - Klassifikation
  - Exkurs: Parametrisch / Nicht-Parametrisch
- Generalisierte Lineare Modelle
- Support Vector Machines (SVM)
- **Neuronale Netzwerke**

Neuronale Netzwerke – Vorbild Biologie

Technische Hochschule Wildau
Technical University of Applied Sciences
WILDAU

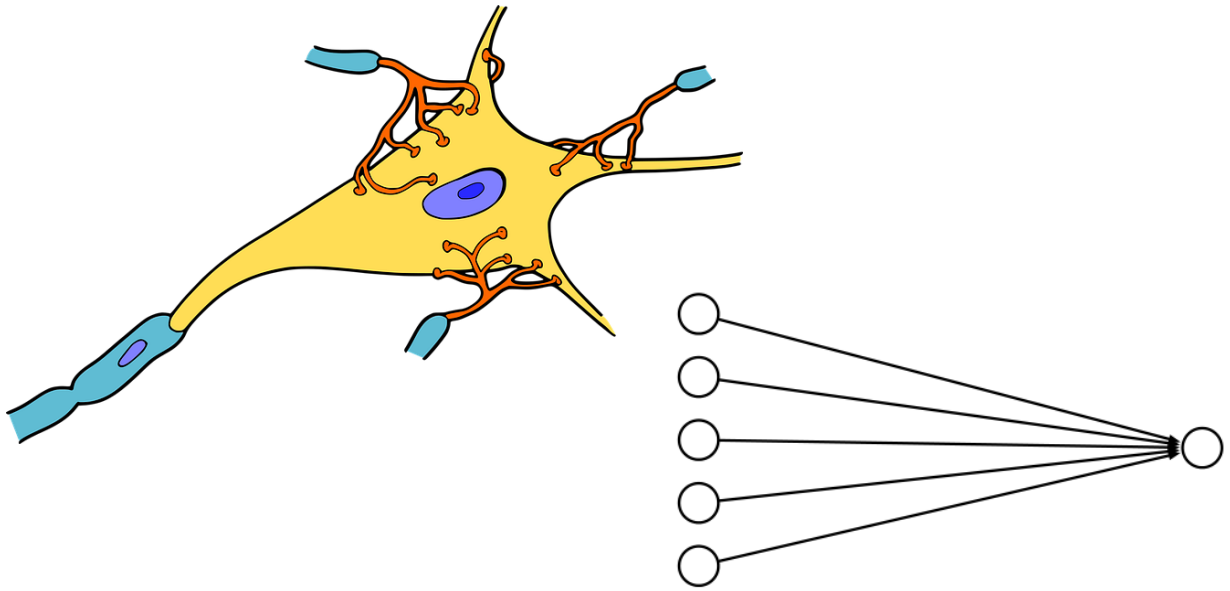
- Kortikale Neuronen
  - Input über Dendriten
  - Output über Axon
  - Durchschnittlich 7000 Verbindungen je Neuron

Quelle: NIH, Wikipedia

Introduction on Neural Networks

Technische Hochschule Wildau
Technical University of Applied Sciences
WILDAU

Single Neuron



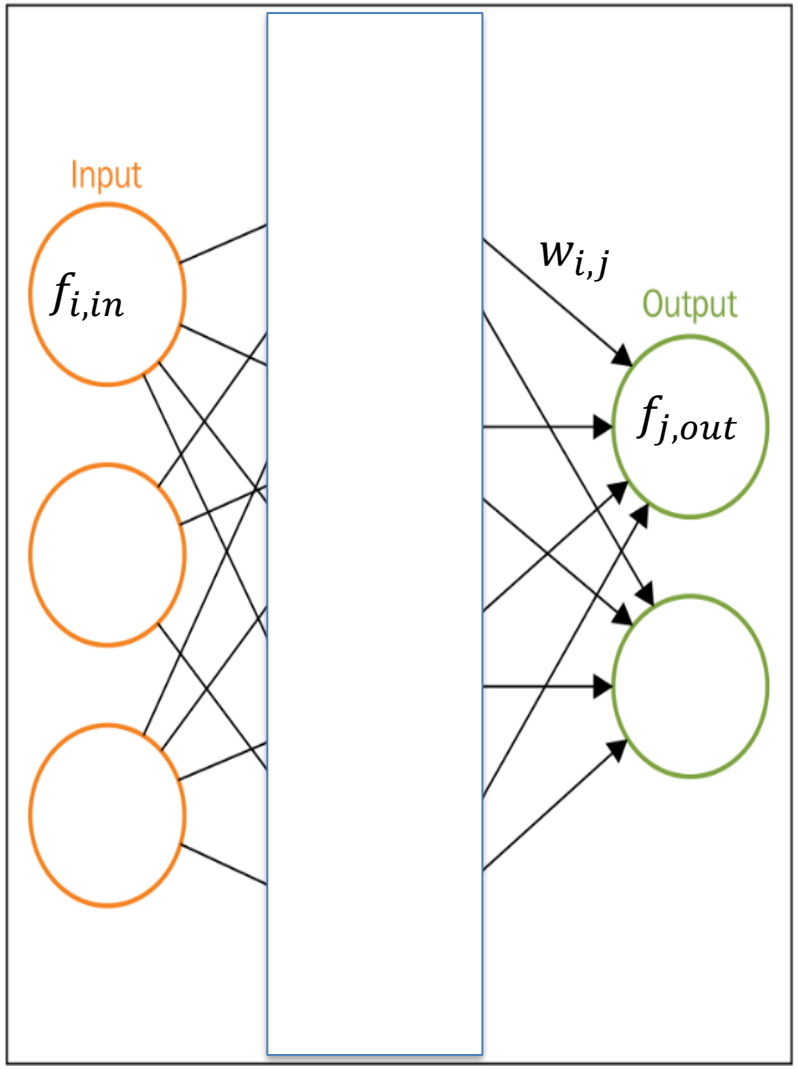
$$\text{score}(\lambda, x) = \sum_j \lambda_j \; h_j(x)$$

= aggregation of weighted features

Rosenblatt (1958), *Perceptron*

Neuronale Netzwerke – Feedforward

- Parallele Vorhersagen
  - y_j: Zielgrößen
- Einlagiges Perzeptron (Rosenblatt, 1958)
  $f_{j,out} = \sum_i w_{i,j} \cdot f_{i,in} = w_j \cdot f_{in}$
  - $f_{i,in}$: Eingabe /Input
  - $w_{i,j}$: Gewichte
  - $f_{j,out}$: Output



Neuronale Netzwerke - Gradientenabstieg

- Anpassung durch Gradienten-abstieg (quadrat. Fehler):

$$L(w_j) = (y_j - f_{j,out})^2$$

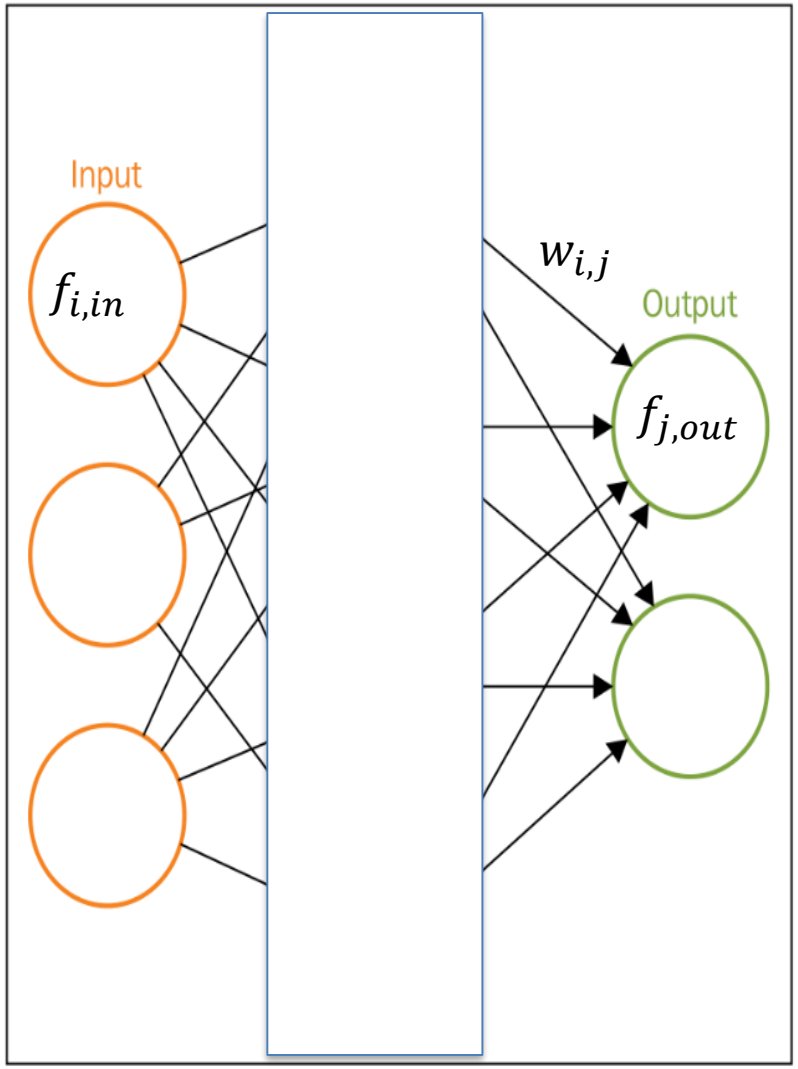
$$= \left(y_j - \sum_i w_{i,j} \cdot f_{i,in}\right)^2$$

$$\frac{\partial}{\partial w_j} L \sim (y_j - f_{j,out}) \cdot f_{i,in}$$

- Update-Regel

$$w_{i,j}^{neu} = w_{i,j}^{alt} + \Delta w_{i,j}$$

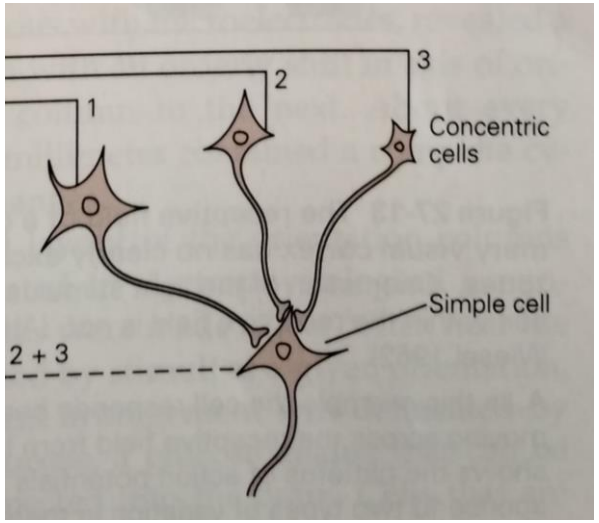
mit $\Delta w_{i,j} = \alpha \cdot (y_j - f_{j,out}) \cdot f_{i,in}$



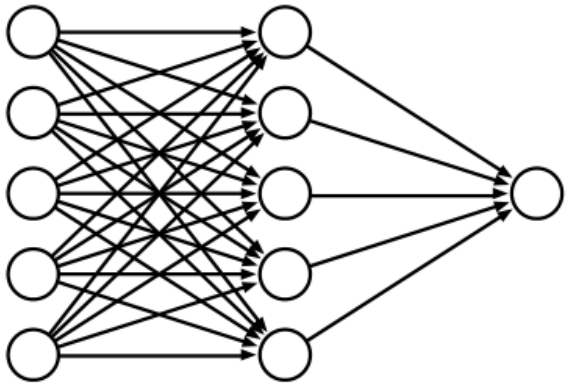
Introduction on Neural Networks

Technische Hochschule Wildau
Technical University of Applied Sciences

Layer of Neurons



Quelle: Kandel et al., Principles of Neural Science



$$h_j = \sum_i x_i w_{ji}$$

$$y_k = \sum_j h_j w_{kj}$$

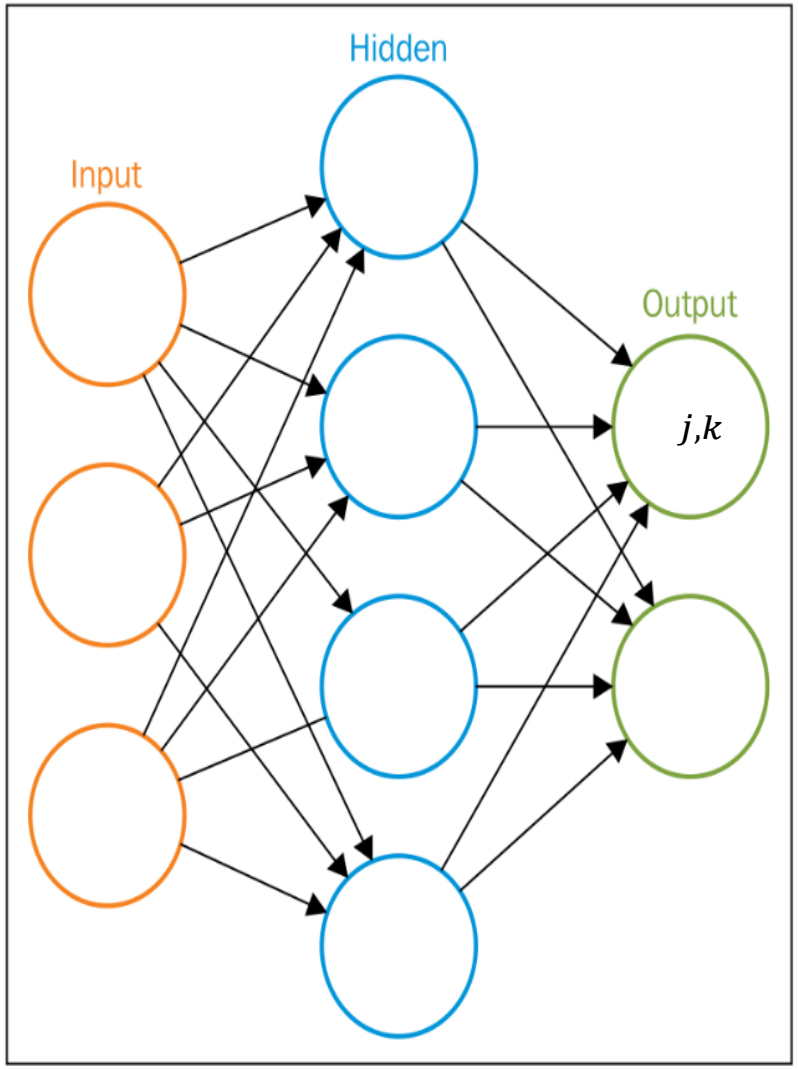
= aggregation of weighted (
aggregation of weighted features
)

Neuronale Netzwerke – Multilayer Feedforward

- Mehrlagiges Perzeptron

$$f_{j,k} = h\left(\sum_{i \in L_{k-1}} w_{i,j,k} \cdot f_{i,k-1}\right)$$

  - k: Schicht / Layer
  - $f_{i,k-1}$: Eingabe /Input
  - $w_{i,j,k}$: Gewichte
  - h: Aktivierungsfunktion
  - $f_{j,k}$: Ausgabe / Output



Neuronale Netzwerke – Multilayer Feedforward

- Anpassung durch Backpropagation = Rückwärtsmeldung des Fehlers:

$$w_{i,j,k}^{neu} = w_{i,j,k}^{alt} + \Delta w_{i,j,k}$$

$$\Delta w_{i,j,k} = \alpha \cdot \delta_{j,k} \cdot f_{i,k-1}$$

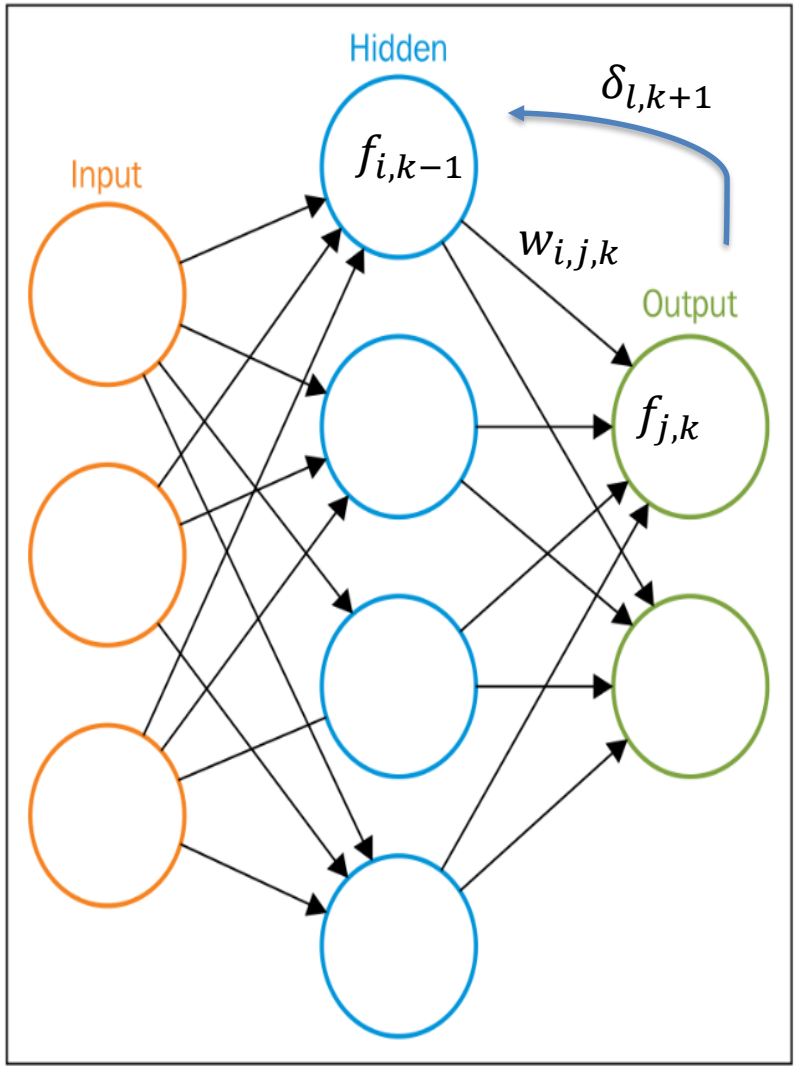
  - Output-Schicht:

$$\delta_{j,out} = (y_j - f_{j,out}) \cdot f'_{j,out}$$

  - Hidden-Schicht:

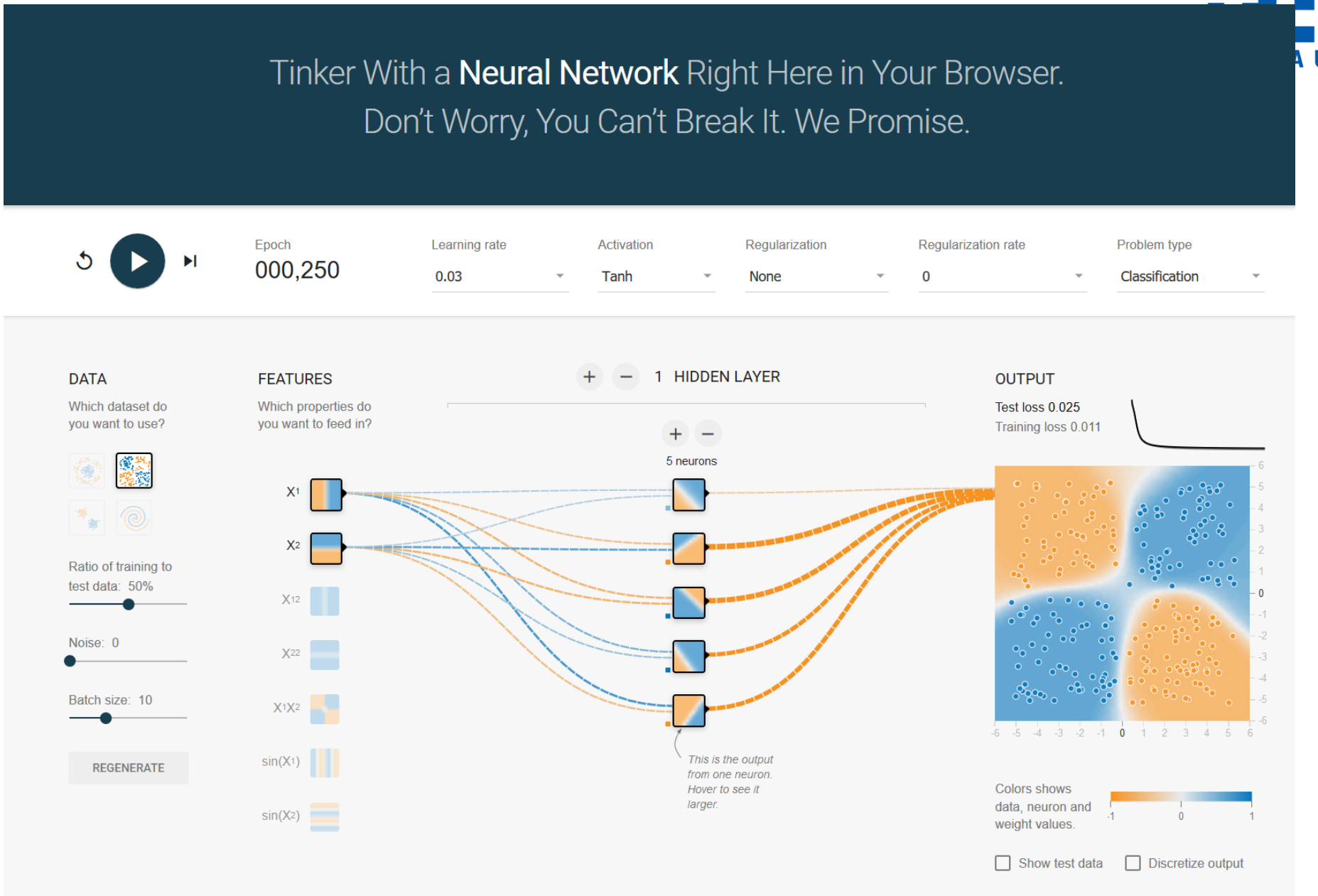
$$\delta_{j,k} = \left(\sum_{l \in L_{k+1}} w_{j,l,k+1} \cdot \boldsymbol{\delta_{l,k+1}}\right) \cdot f'_{j,k}$$

$\boldsymbol{\delta_{l,k+1}}$ ist der rückwärtsgemeldete Fehler



playground.tensorflow.org

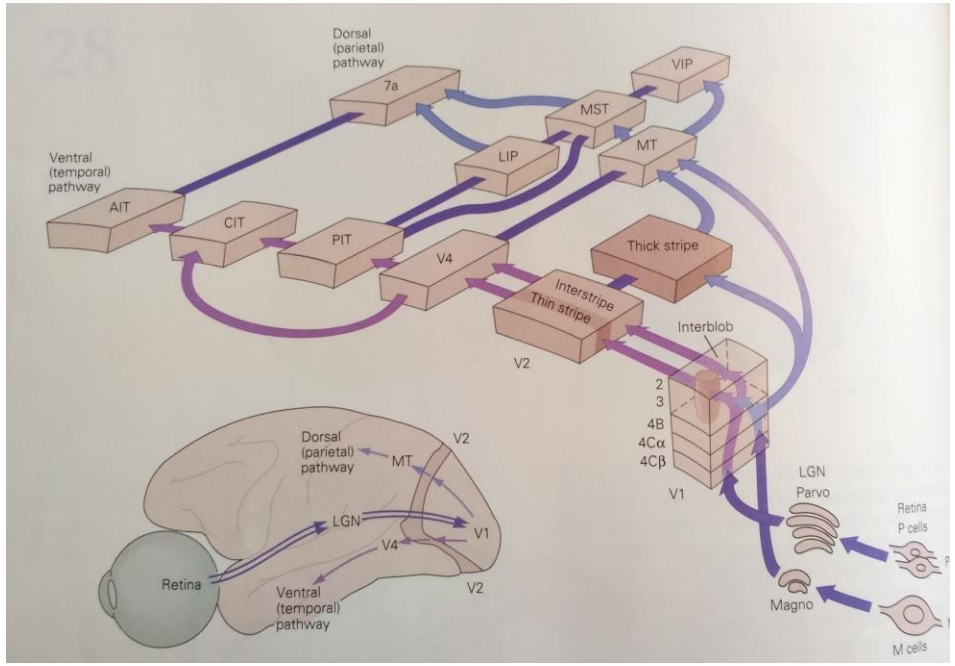
Technische Hochschule Wildau
Technical University of Applied Sciences



Neuronale Netzwerke – Vorbild Biologie

- Kortikale Neuronen
  - Input über Dendriten
  - Output über Axon
  - Durchschnittlich 7000 Verbindungen je Neuron

- Visueller Cortex
  - Netzwerk von Neuronen
  - Hierarchische Struktur

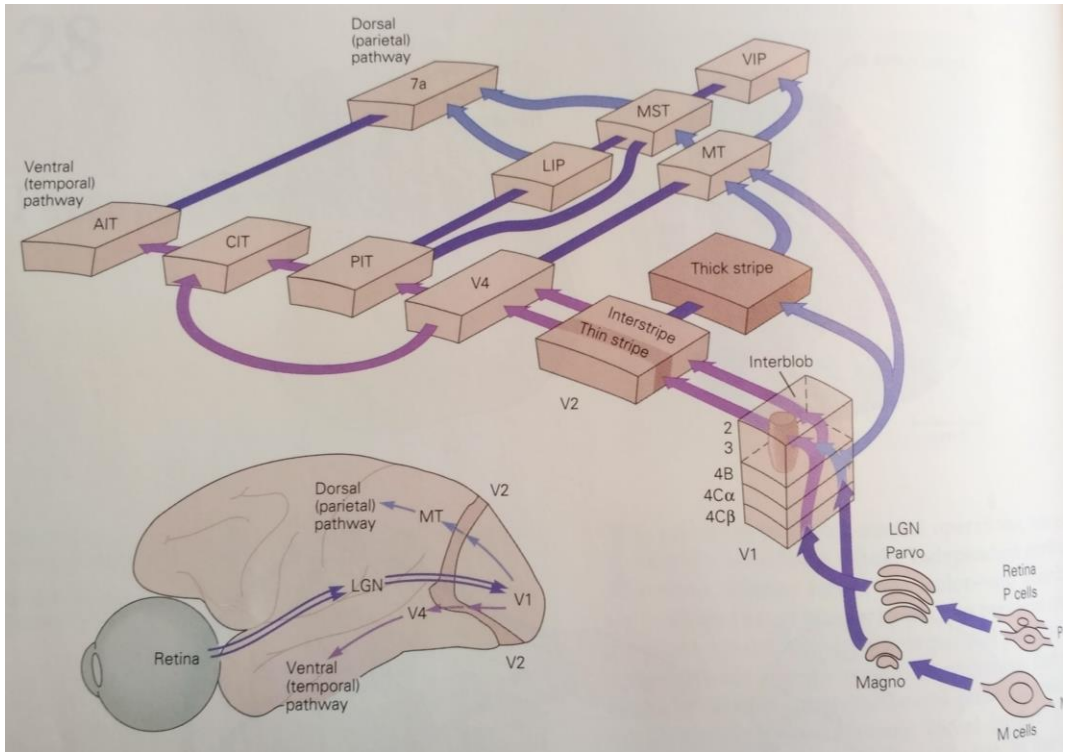


Quelle: Kandel et al., Principles of Neural Science

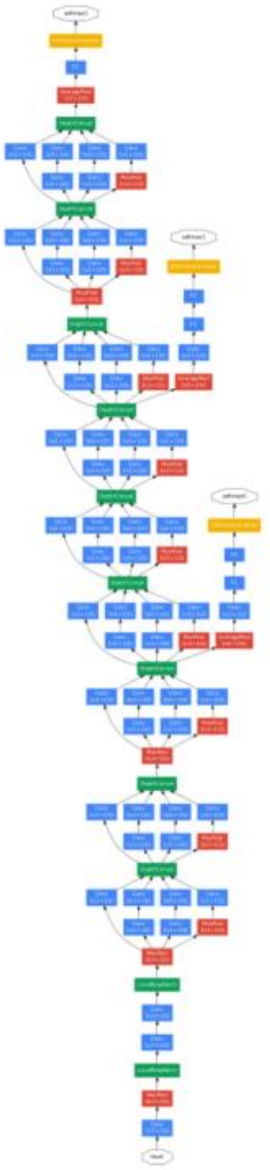
Introduction on Neural Networks

Technische Hochschule Wildau
Technical University of Applied Sciences
WILDAU

Deep Neural Network



Quelle: Kandel et al., Principles of Neural Science



Quelle: Szegedy et al., 2015

= aggregation of weighted(...)

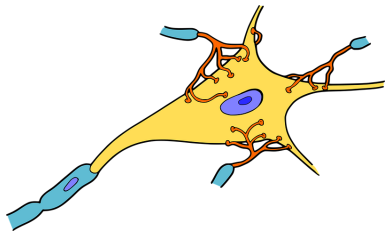
= AWF = aggregation of weighted AWF

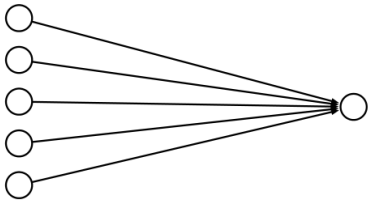
= universal approximator (Hornik, Stinchcombe, White, 1989)

Introduction on Neural Networks

Technische Hochschule Wildau
Technical University of Applied Sciences

Single Neuron

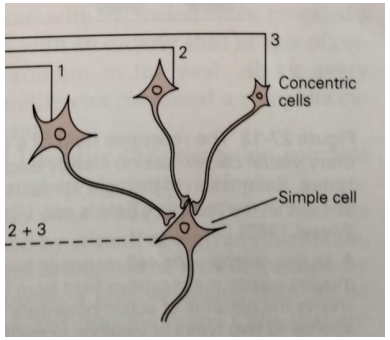


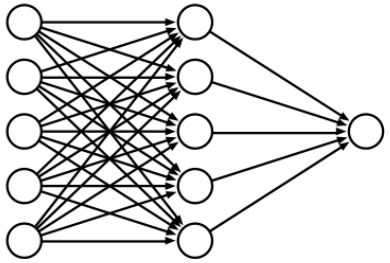


$$\text{score}(\lambda, x) = \sum_j \lambda_j \; h_j(x)$$

= aggregation of weighted features

Multilayer Feedforward





$$h_j = \sum_i x_i w_{ji}$$

$$y_k = \sum_j h_j u_{kj}$$

= aggregation of weighted (
aggregation of weighted features
)

Deep Learning