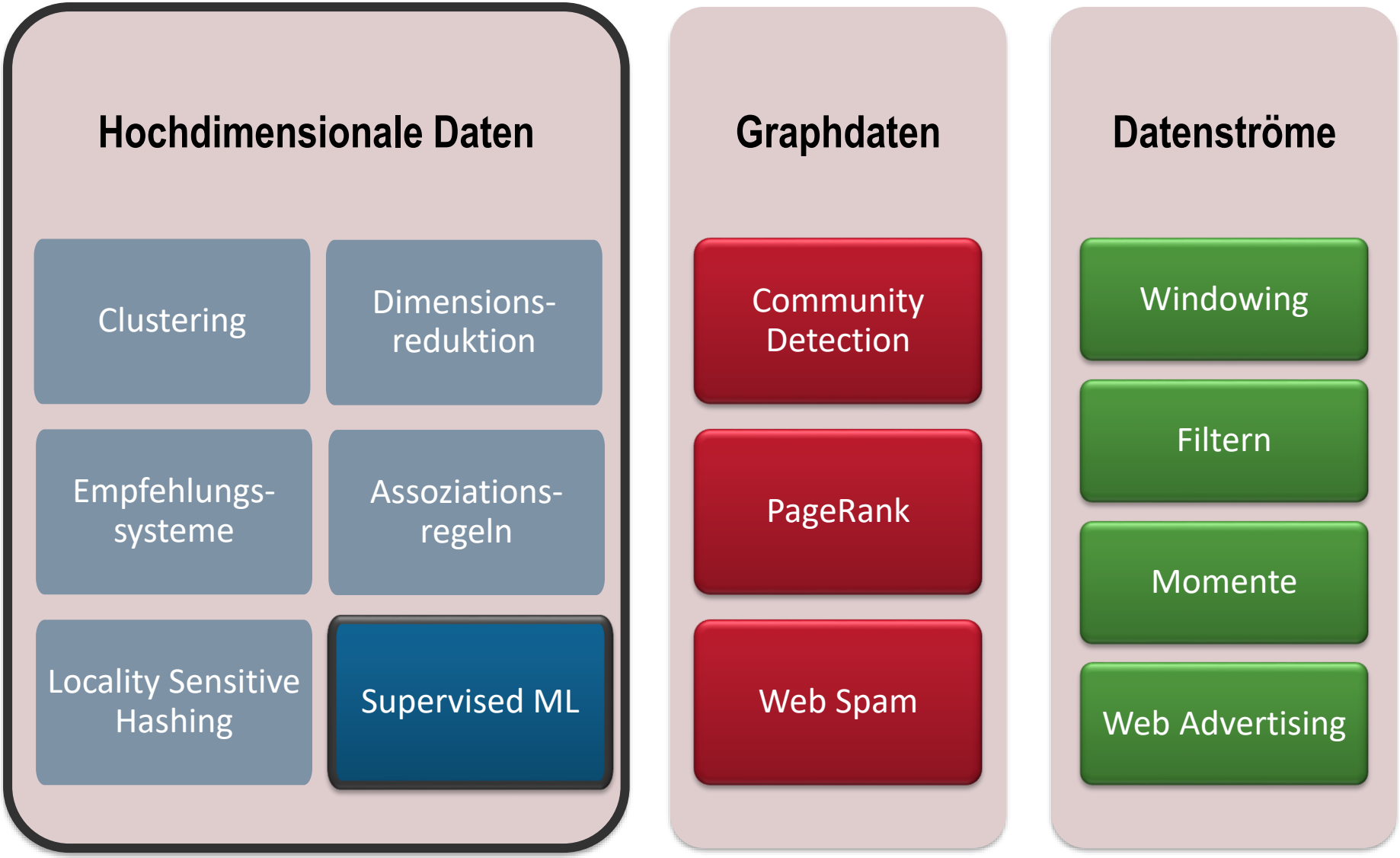
Data Mining

Supervised Machine Learning

Johannes Zschache
Wintersemester 2019

Abteilung Datenbanken, Universität Leipzig
http://dbs.uni-leipzig.de

Übersicht


Hochdimensionale Daten
Clustering
Dimensions-reduktion
Empfehlungs-systeme
Assoziations-regeln
Locality Sensitive Hashing
Supervised ML
Graphdaten
Community Detection
PageRank
Web Spam
Datenströme
Windowing
Filtern
Momente
Web Advertising

Inhaltsverzeichnis

Literatur: Kapitel 12 und 13 aus „Mining of Massive Datasets“ (**v3.0 beta**): http://www.mmds.org

Supervised Learning

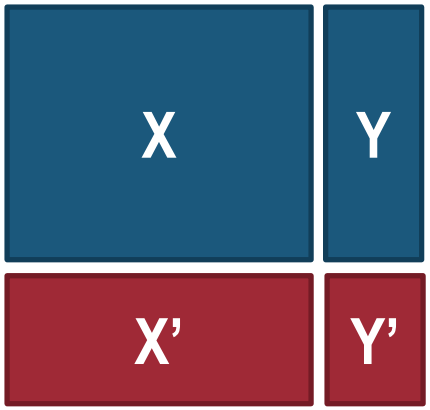
- Daten zu einem Paar $(\boldsymbol{x}, y)$
  - $\boldsymbol{x}$ ist ein Vektor aus Merkmalen (Features):
    - verschiedene Datentypen möglich
    - Beispiel: (Age, Male, Class)
  - y ist eine Bezeichnung (Label),
    - eine reelle Zahl oder eine Kategorie (Klasse)
    - Beispiel: Survived

Age	Male	Class	Survived
Child	True	1	Yes
Adult	True	2	No
Adult	False	1	Yes
Child	True	3	No
…	…	…	…

- **Ziel**: Auffinden einer Funktion (Modell) f mit

$$y = f(\boldsymbol{x})$$

- Vielzahl an Funktionen möglich
- Bewertung und Auswahl der Funktion über Daten
  - Aufteilung in Trainings- und Testdaten (z.B. 80% vs. 20%)
  - Schätzen der Funktion über Trainingsdaten: *Fehler möglichst gering*
  - Bewerten der Funktion an Testdaten: *Fehler möglichst gering*



Trainings-/ Testdaten

Methoden des Supervised ML

Bewährte Methoden (Ausschnitt)
Lineare Regression
Logistische Regression
Linear Discriminant Analysis
K-Nearest Neighbors
Naïve Bayes
Entscheidungsbäume
Support Vector Machines
Neuronale Netze

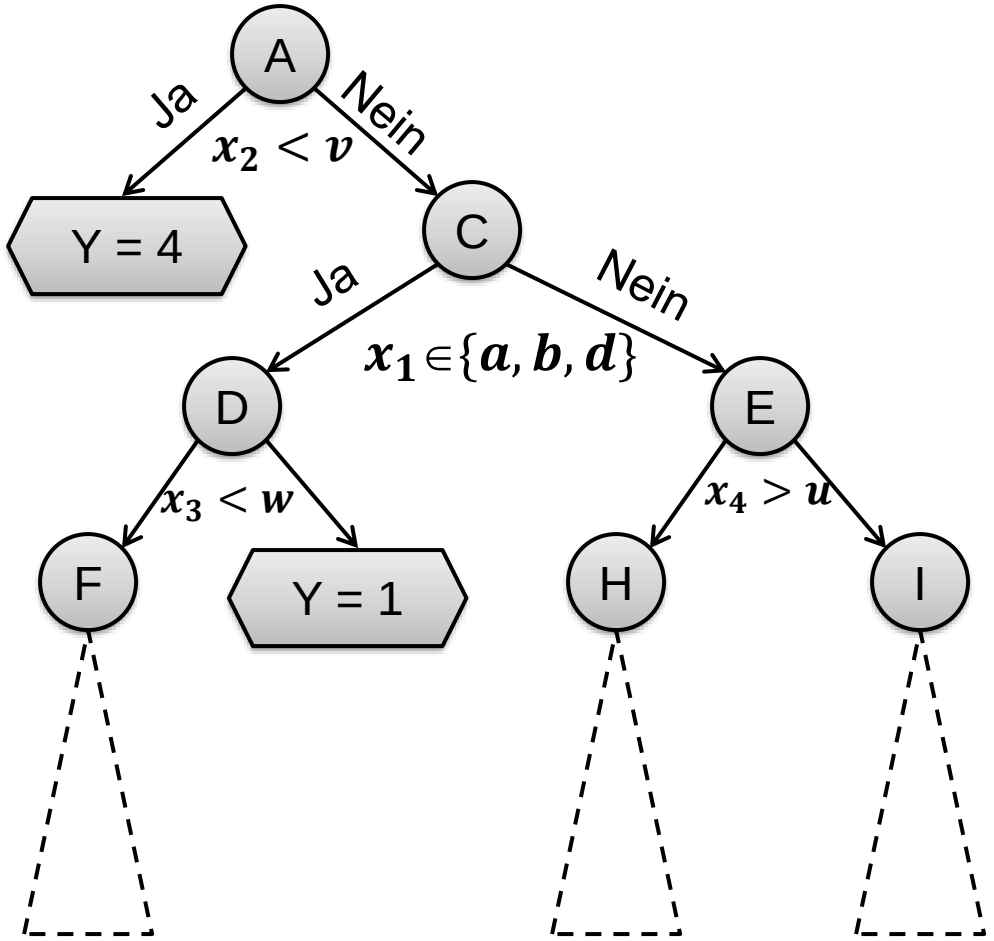
- Generelles Problem bei hochdimensionalen Daten: **Overfitting** (Funktion passt sich den zufälligen Fehlern an)
- Lösung z.B.
  - Dimensionsreduktion über PCA/SVD oder Clustering
  - Regularisierung (Ridge, Lasso)
  - Informationskriterien, z.B. AIC, BIC, DIC, WAIC, …

Inhaltsverzeichnis

Literatur: Kapitel 12 und 13 aus „Mining of Massive Datasets“ (**v3.0 beta**): http://www.mmds.org

Entscheidungsbäume

- Variable Y: numerisch (Regression) oder kategorial (Klassifikation)
- Vorhersage von Y durch X über eine Baumstruktur
  - Folge den Ästen des Baums entsprechend den Werten von X
  - Vorhersagewert für Y an den Blattknoten



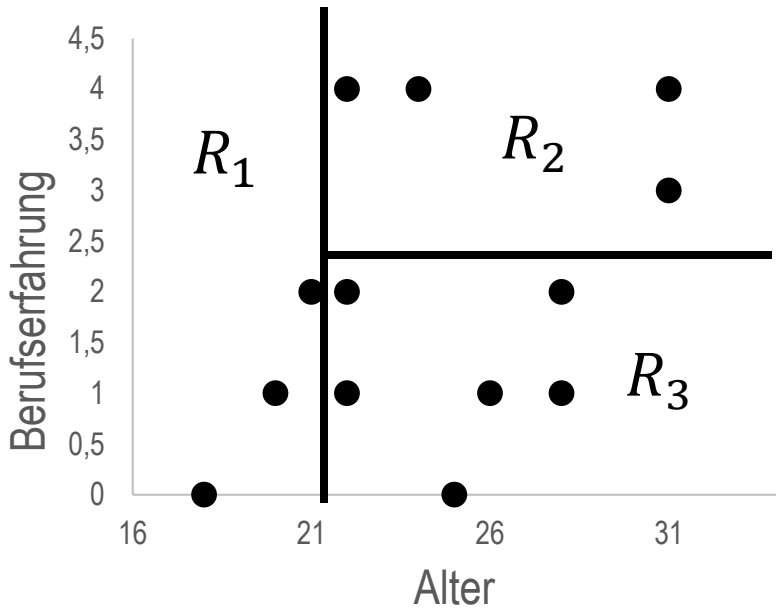
Erstellen eines Entscheidungsbaums (Regression)

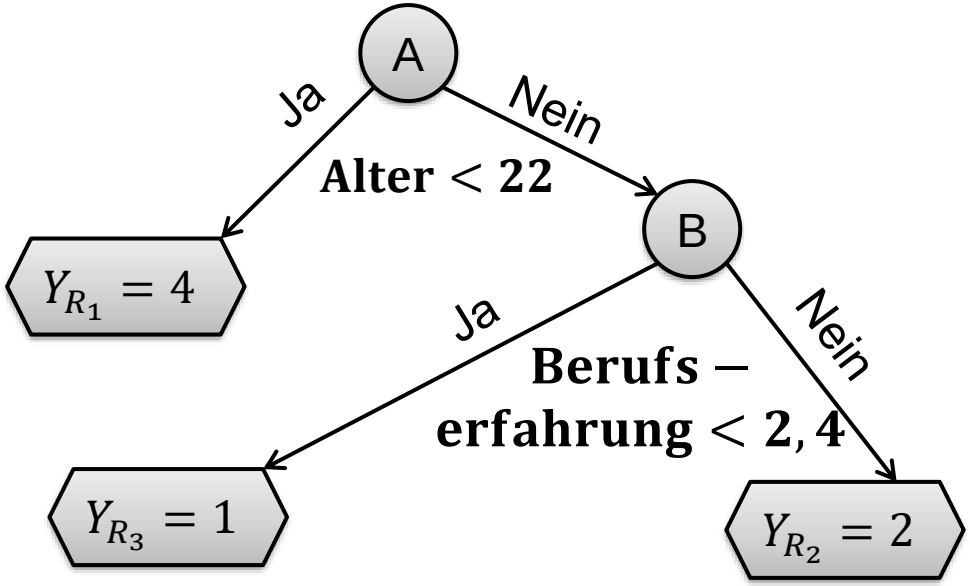
Ziel: Aufteilung der Variaben aus X in Regionen $R_1, R_2, \ldots, R_J$, so dass

$$\sum_{j=1}^{J} \sum_{i \in R_j} \left(y_i - \widehat{y_{R_j}} \right)^2$$

minimiert wird ($\widehat{y_{R_j}}$ bezeichnet den Vorhersagewert der Region R_j).

- Vorhersagewert ist der *Mittelwert* der Datenpunkte der Region





Erstellen eines Entscheidungsbaums (Regression)

Ziel: Aufteilung der Variaben aus X in Regionen $R_1, R_2, \ldots, R_J$, so dass

$$\sum_{j=1}^{J} \sum_{i \in R_j} \left(y_i - \widehat{y_{R_j}} \right)^2$$

minimiert wird ($\widehat{y_{R_j}}$ bezeichnet den Vorhersagewert der Region R_j).

- Vorhersagewert ist der *Mittelwert* der Datenpunkte der Region
- **Problem**: zu viele mögliche Aufteilungen in J Regionen
- Ausweg: **Recursive Binary Splitting**
  - Wiederholte Auswahl einer Variable X_j und eines Schwellenwerts s, so dass die beiden Regionen $R_1(j,s) = \{x | x_j < s\}$ und $R_2(j,s) = \{x | x_j \geq s\}$ die folgende Summe minimieren:

$$\sum_{i: x_i \in R_1(j,s)} \left(y_i - \widehat{y_{R_1}} \right)^2 + \sum_{i: x_i \in R_2(j,s)} \left(y_i - \widehat{y_{R_2}} \right)^2$$

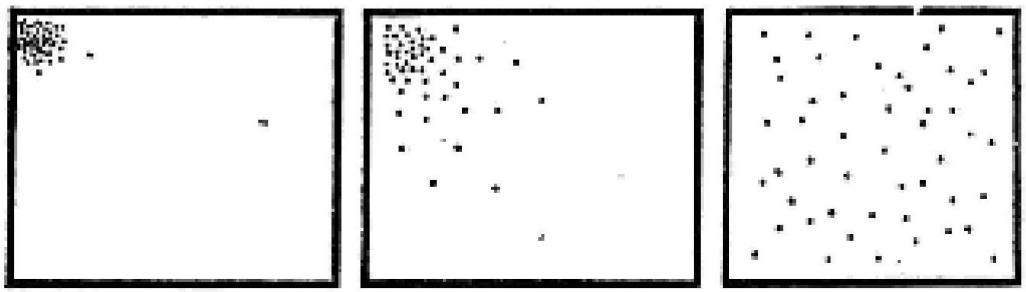
  - Stopp, falls z.B. in jede Region weniger als 5 Datenpunkte fallen

Entscheidungsbaum für Klassifikation

- Vorhersagewert ist der *häufigste Wert* der Datenpunkte einer Region
- Fehlermaß: z.B. **Entropie**

$$\boldsymbol{H}(Y) = -\sum_{y \in \text{Domäne von } Y} p_y \log_2 p_y$$

- Der Wert p_y gibt die Wahrscheinlichkeit der Ausprägung $Y = y$
  - Hohe Entropie: gleichverteiltes Y
  - Niedrige Entropie: ungleiche Verteilung



Niedrige Entropie **Hohe Entropie**

Entropie: Beispiel

- $\boldsymbol{X}$: Abschluss
- $\boldsymbol{Y}$: mag den Film “Casablanca”
- Schätzen der Wahrscheinlichkeiten über relative Häufigkeiten
  - $P(Y = Ja) = \frac{3}{8}, P(Y = Nein) = \frac{5}{8}$
  - $P(Y = Ja|X = Mathe) = \frac{1}{4}$
  - $P(Y = Nein|X = Mathe) = \frac{3}{4}$

X	Y
Mathe	Ja
Geschichte	Nein
Informatik	Ja
Mathe	Nein
Mathe	Nein
Informatik	Ja
Mathe	Nein
Geschichte	Nein

- Entropie:

$$H(Y) = -\frac{3}{8}\log_2\frac{3}{8} - \frac{5}{8}\log_2\frac{5}{8} \approx 0.95$$

- Spezifische bedingte Entropie:

$$H(Y|X = Mathe) = -\frac{1}{4}\log\frac{1}{4} - \frac{3}{4}\log\frac{3}{4} \approx 0.81$$

Entropie: Beispiel

- Bedingte Entropie:

$$H(Y|X) = \sum_{x \in O_X} P(X = x) H(Y|X = x)$$

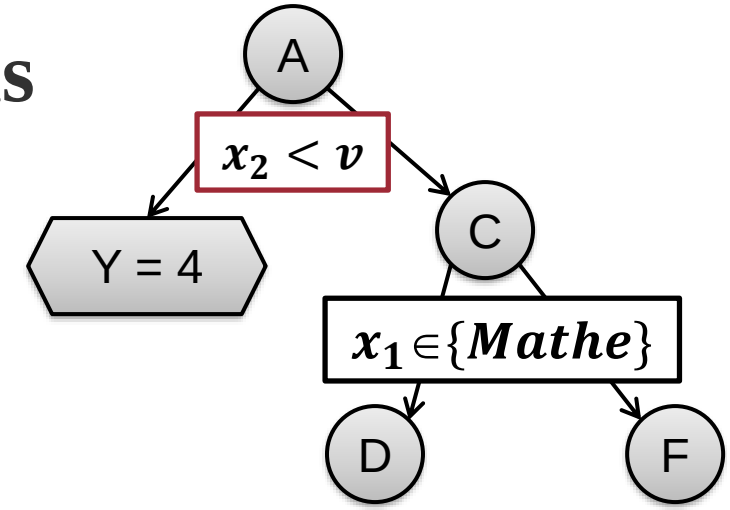
x	$P(X = x)$	$H(Y|X = x)$
Mathe	$\frac{1}{2}$	0.81
Geschichte	$\frac{1}{4}$	0
Informatik	$\frac{1}{4}$	0

X	Y
Mathe	Ja
Geschichte	Nein
Informatik	Ja
Mathe	Nein
Mathe	Nein
Informatik	Ja
Mathe	Nein
Geschichte	Nein

$$H(Y|X) = \frac{1}{2} \cdot 0.81 + \frac{1}{4} \cdot 0 + \frac{1}{4} \cdot 0 = 0.4$$

- **Ziel:** Auswahl des Attributs X mit niedrigstem $H(Y|X)$

Beste Aufteilung eines Knotens



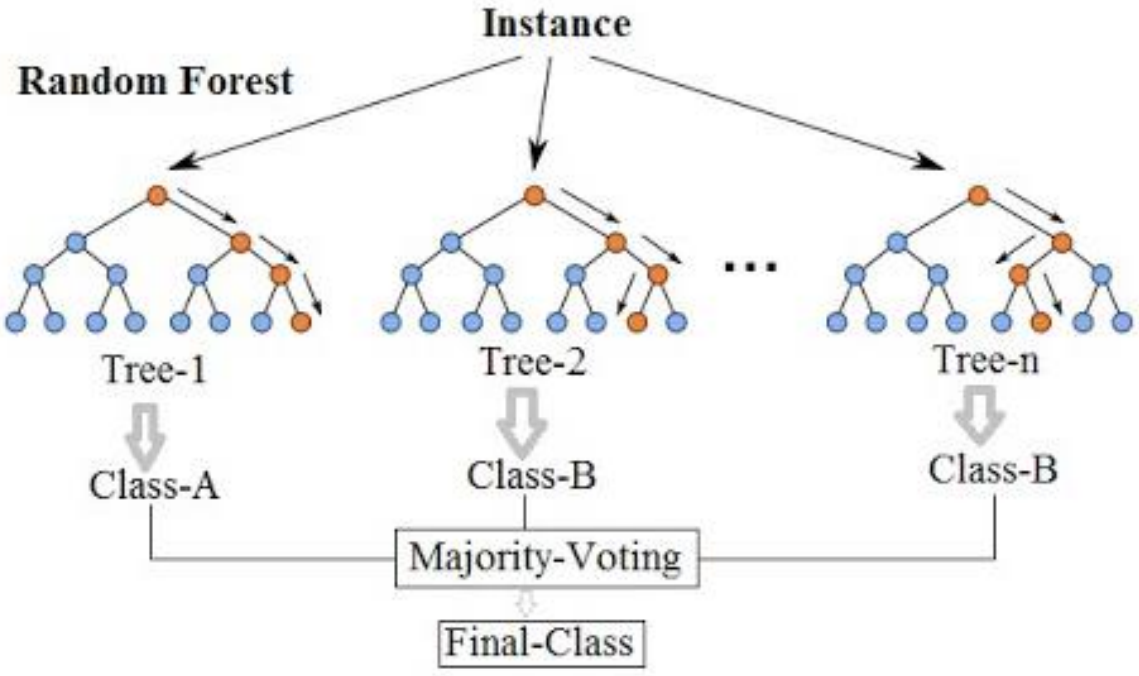
- Anschließend
  - X ist numerisch: Auswahl des Schwellenwertes
  - X ist kategorisch: Auswahl von Kategorien
- Beispiel:
  - Aufteilung: $X = Mathe$ vs. $X \neq Mathe$
    - $H(Y|X = Mathe) = 0.81$ und $H(Y|X \neq Mathe) = 1$
    - Spezifische bedingte Entropie gewichtet nach der Anzahl der Einträge pro Kindsknoten:
      $$\frac{1}{2} \cdot H(Y \mid X = Mathe) + \frac{1}{2} H(Y|X \neq Mathe) = 0.9$$
  - **Aufteilung: $X = Informatik$ vs. $X \neq Informatik$**
    - $H(Y|X = Informatik) = 0$ und $H(Y|X \neq Informatik) = 0.65$
    - $\frac{1}{4} \cdot H(Y|X = Informatik) + \frac{3}{4} H(Y|X \neq Informatik) = \mathbf{0.48}$
  - Aufteilung: $X = Geschichte$ vs. $X \neq Geschichte$
    - $H(Y|X = Geschichte) = 0$ und $H(Y|X \neq Geschichte) = 1$
    - $\frac{1}{4} \cdot H(Y|X = Geschichte) + \frac{3}{4} H(Y|X \neq Geschichte) = \frac{3}{4}$

Entscheidungsbäume

- Leicht zu verstehen, implementieren und interpretieren
- Parallelisierbar:
  - B. Panda, J. S. Herbach, S. Basu, and R. J. Bayardo. ***PLANET: Massively parallel learning of tree ensembles with MapReduce.*** In Proc. *VLDB* 2009.
  - J. Ye, J.-H. Chow, J. Chen, Z. Zheng. ***Stochastic Gradient Boosted Distributed Decision Trees***. In Proc. *CIKM* 2009.
- Sowohl für kategoriale als auch metrische Ergebnisvariable Y geeignet
- Problem: *Overfitting* (Überanpassung des Modells an die Daten)
  - Overfitting bei zu vielen Ebenen
  - Doch bei wenigen Ebenen können nur wenige Attribute verwendet werden und die Vorhersagegenauigkeit ist oft gering

Bagging und Random Forests

- Ausweg: Kombination mehrerer Entscheidungsbäume geringer Tiefe
- Über z.B. *Bagging*
  - Ziehen mehrerer Zufallsstichproben aus den Daten (mit Zurücklegen)
  - Ein Entscheidungsbaum geringer Tiefe pro Stichprobe
  - Mittelwert/häufigster Wert über alle Bäume ergibt Vorhersage
- *Random Forest*:
  Zusätzlich zum Bagging wird beim Lernen der Bäume an jedem Knoten nur eine kleine (zufällige) Auswahl der Attribute betrachtet

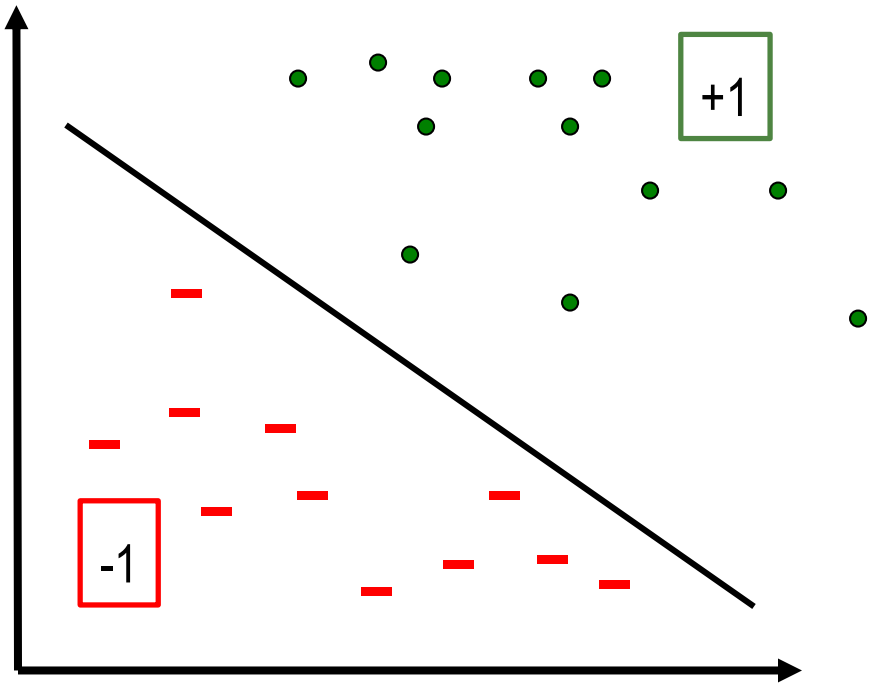


Inhaltsverzeichnis

Literatur: Kapitel 12 und 13 aus „Mining of Massive Datasets“ (**v3.0 beta**): http://www.mmds.org

Klassifizierung über Hyperebene

- Numerischer Merkmalsvektor $\boldsymbol{x} = (x_1, \ldots x_n)$
- Binäre Variable $y \in \{-1, +1\}$



Klassifizierung über Hyperebene

- Numerischer Merkmalsvektor $\boldsymbol{x} = (x_1, \ldots x_n)$
- Binäre Variable $y \in \{-1, +1\}$

- Eine Hyperebene des $\mathbb{R}^n$ teilt diesen Raum in 2 Bereiche
- Die Gewichte $\boldsymbol{w} = (w_0, w_1, \ldots, w_n) \in \mathbb{R}^{n+1}$ beschreiben eine *Hyperebene H* über

$$H = \left\{ x \in \mathbb{R}^n | w_0 + \sum_{i=1}^{n} w_i \, x_i = 0 \right\}$$

- Klassifizierung über Hyperebene:

$$\hat{y} = +1, \text{ falls } w_0 + \textstyle\sum_i w_i x_i > 0$$
$$\hat{y} = -1, \text{ falls } w_0 + \textstyle\sum_i w_i x_i < 0$$

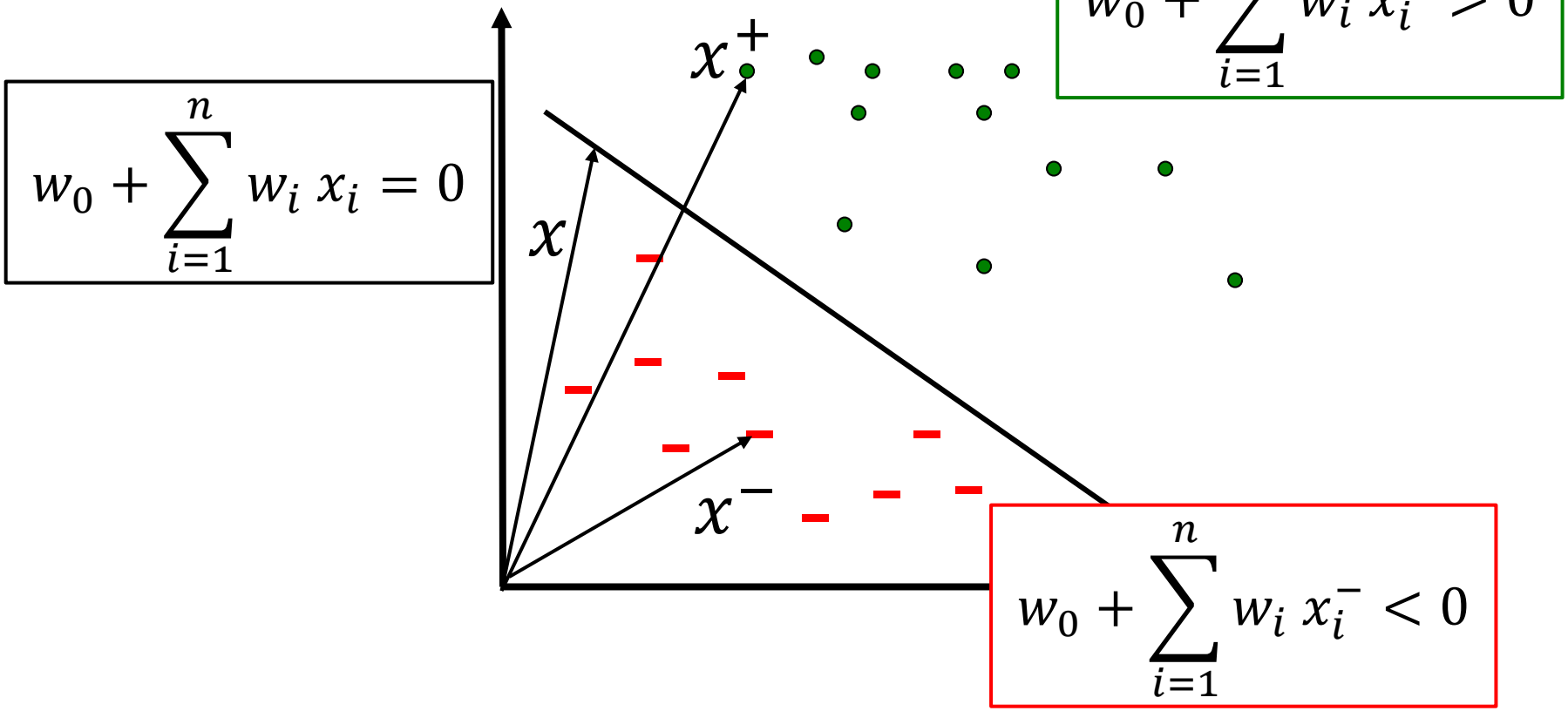
Klassifizierung über Hyperebene

Ziel: Finden der Parameter $\boldsymbol{w}$, so dass der Raum der Merkmalsvektoren in zwei Teile aufgespalten wird und Punkte mit dem gleichen Label auf der gleichen Seite sind

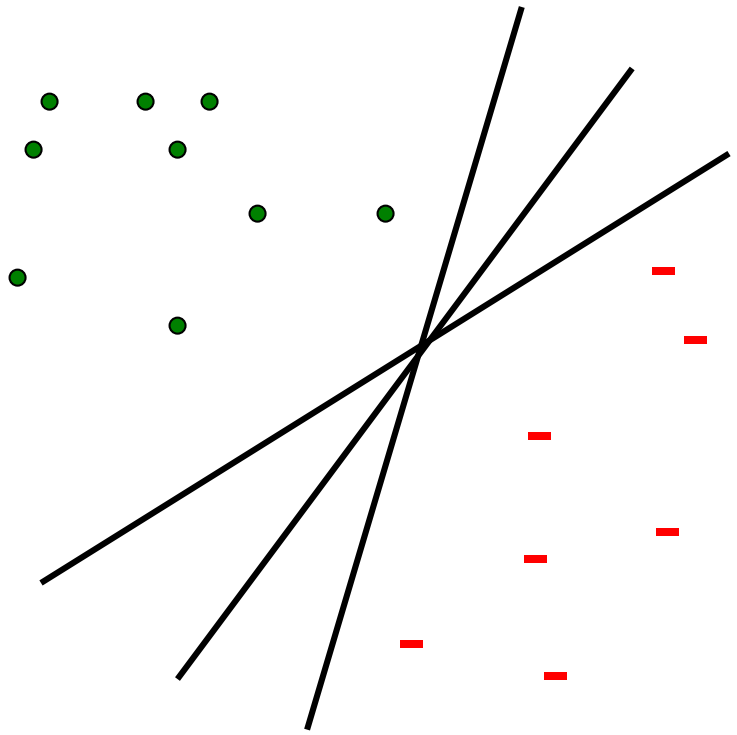
$$w_0 + \sum_{i=1}^{n} w_i \, x_i = 0$$

$$w_0 + \sum_{i=1}^{n} w_i \, x_i^+ > 0$$

$$w_0 + \sum_{i=1}^{n} w_i \, x_i^- < 0$$



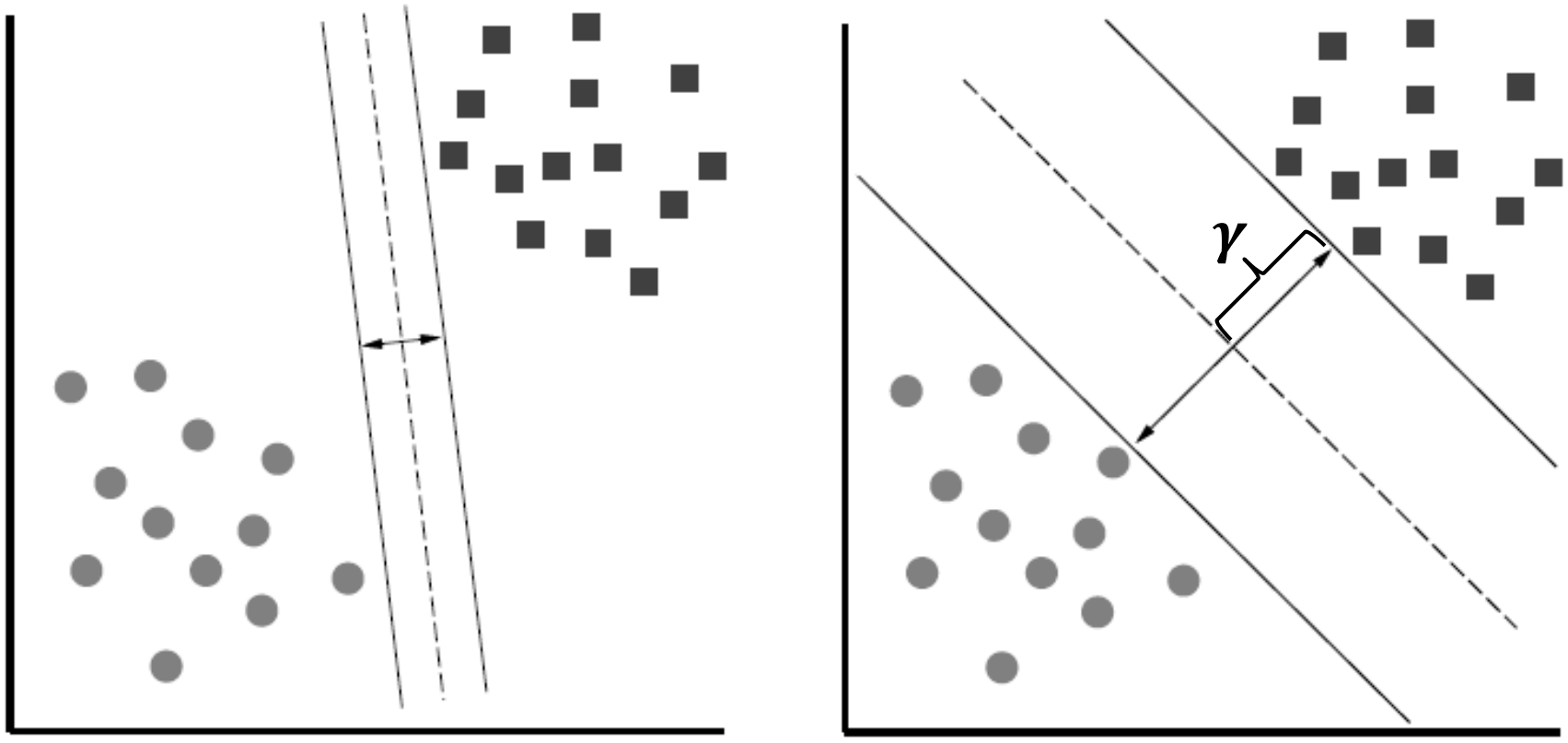
Auswahl der Hyperebene



Welche Hyperebene ist die beste?

Maximal Margin Classifier

Verwendung der Hyperebene mit maximalen Abstand γ zu den Daten



Maximal Margin Classifier

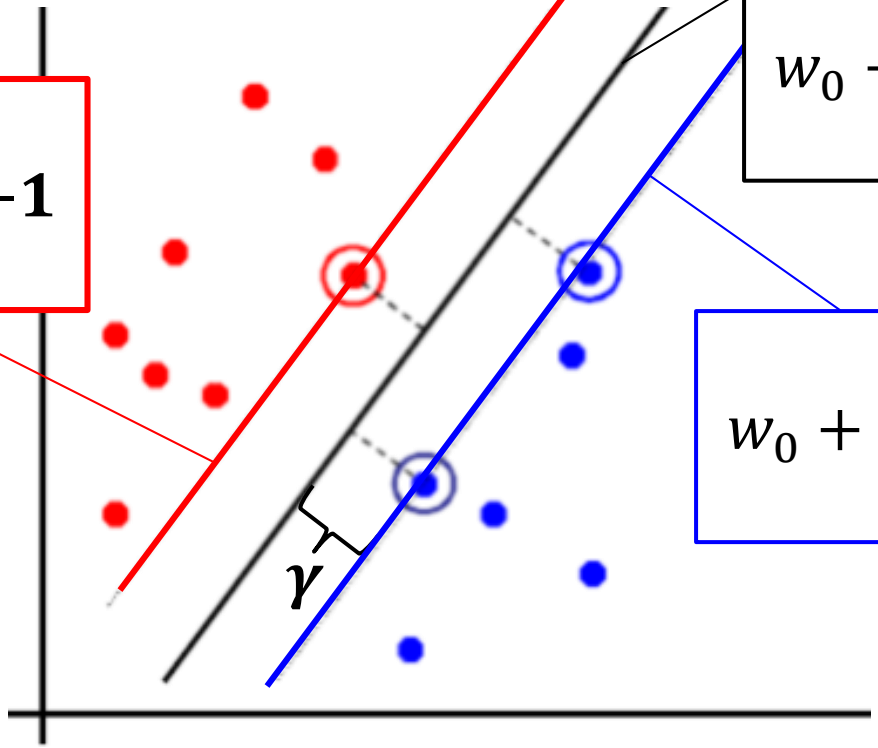
Ziel: Suche nach Gewichten $\boldsymbol{w} = (w_0, w_1, \ldots, w_n)$ mit **minimalem Wert** für $\sum_{i=1}^{n} \mathbf{w_i^2}$, so dass, für alle Daten $(\boldsymbol{x}, y)$, gilt:

$$y\left(w_0 + \sum_{i=1}^{n} w_i\, x_i\right) \geq 1$$

$$w_0 + \sum_{i=1}^{n} w_i\, x_i = -1$$

$$w_0 + \sum_{i=1}^{n} w_i\, x_i = 0$$

$$w_0 + \sum_{i=1}^{n} w_i\, x_i = +1$$

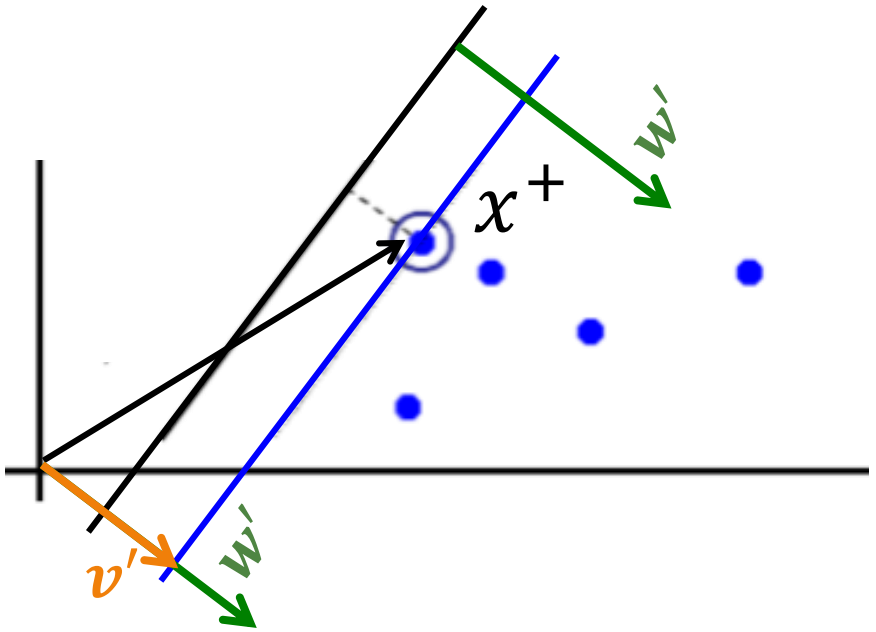


Maximal Margin Classifier

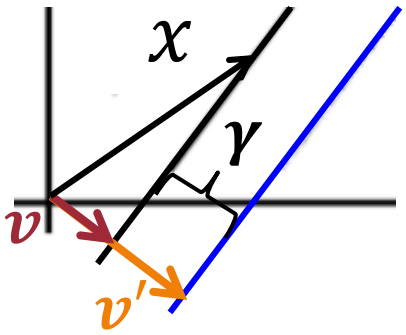
Sei $\mathbf{w}' = (w_1, \ldots, w_n)$. Für $\boldsymbol{x}^+$ gilt:

$$
\begin{aligned}
1 &= w_0 + \sum_{i=1}^{n} w_i \, x_i^+ \\
&= \mathrm{w}_0 + \mathbf{w}' \cdot \boldsymbol{x}^+ \\
&= w_0 + |\mathbf{w}'||\boldsymbol{v}'| \\
&= \mathrm{w}_0 + |\mathbf{w}'|(|\boldsymbol{v}| + \gamma) \\
&= \mathrm{w}_0 + |\boldsymbol{w}'||\boldsymbol{v}| + |\boldsymbol{w}'|\gamma \\
&= \mathrm{w}_0 + \boldsymbol{w}' \cdot \boldsymbol{x} + |\boldsymbol{w}'|\gamma \\
&= 0 + |\boldsymbol{w}'|\gamma
\end{aligned}
$$

Also: $1 = |\boldsymbol{w}'|\gamma$



Orthogonale Projektion von x^+ auf w'



Rand γ ist maximal, wenn

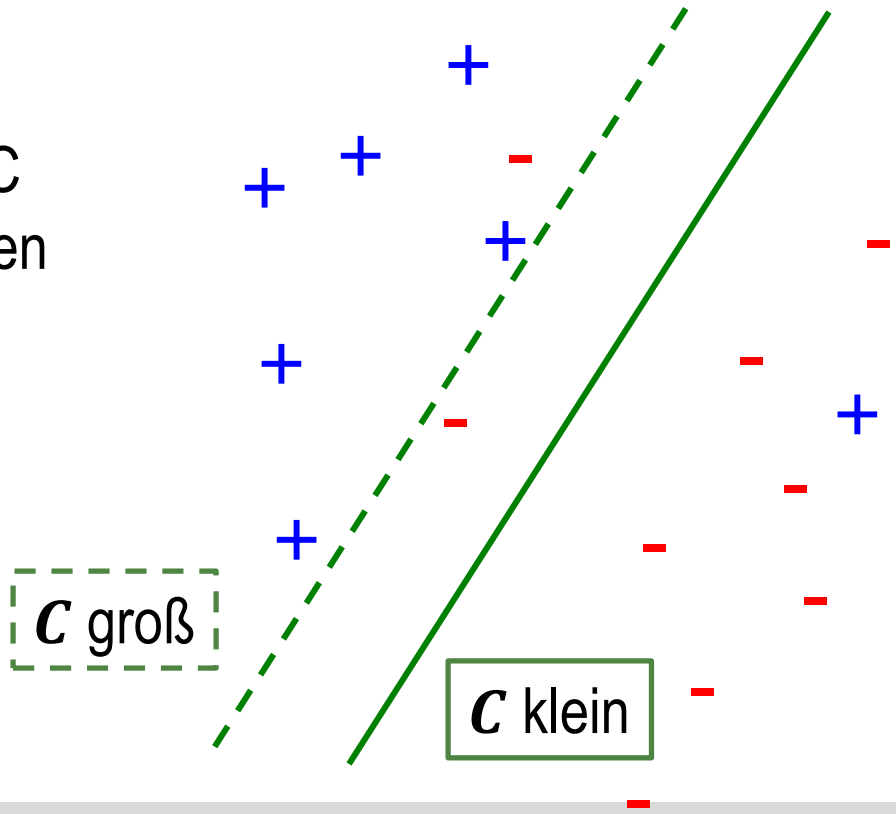
$|\boldsymbol{w}'| = \sqrt{\sum_{\mathbf{i=1}}^{\mathbf{n}} \mathbf{w_i^2}}$ minimal

Linear nicht trennbare Daten

- Falls Daten nicht linear trennbar, Einführung einer *Bestrafung* für falsche Zuordnungen:

$$\min_{w} \sum_{i=1}^{n} w_i^2 + C \cdot (\#\ \textbf{falsche Zuordnungen})$$

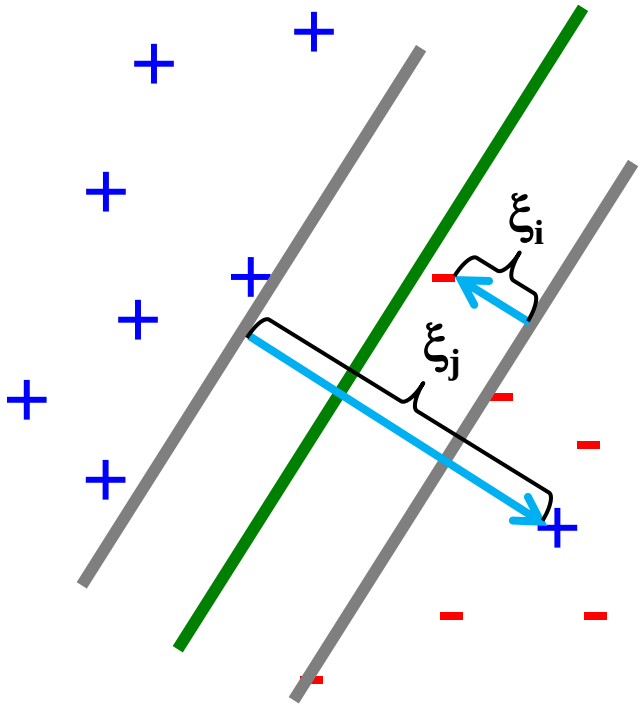
- Optimaler Wert für den Parameter C kann über Testdaten ermittelt werden
  - C groß: wichtig ist die Trennung der Daten (soweit möglich)
  - C klein: wichtig ist ein großer Rand



Linear nicht trennbare Daten

- Nicht alle falschen Zuordnungen sind gleich wichtig
- Bestrafung ξ_i

$$\min_{\boldsymbol{w}} \sum_{\mathbf{i=1}}^{\mathbf{n}} \mathbf{w_i^2} + \boldsymbol{C} \cdot \sum_{j} \xi_j$$



- *Hinge Loss* für Datenpunkt $(\boldsymbol{x}_i, y_i)$:

$$\xi_j := \max\left(0{,}1 - y_j\left(w_0 + \sum_{i=1}^{n} w_i\, x_{ji}\right)\right)$$

  - $\xi_j = 0$ falls x_j auf “richtiger Seite” der Ränder
  - $\xi_j \leq 1$ falls x_j auf “richtiger Seite” der Hyperebene
  - $\xi_j > 1$ falls x_j auf “falschen Seite” der Hyperebene

Support Vector Classifier

- **Zielfunktion**:

$$\min_{\boldsymbol{w}} \sum_{\mathbf{i=1}}^{\mathbf{n}} \mathbf{w_i^2} + \boldsymbol{C} \cdot \sum_{j} \max\left(0{,}1 - y_j \left(w_0 + \sum_{i=1}^{n} w_i \, x_{ji}\right)\right)$$

- Schätzung der Parameter: (Stochastic) Gradient Descent

- Als ***Support-Vektoren*** bezeichnet man die Punkte des Datensatzes, welche direkt auf den Rändern oder auf der falschen Seite des zugehörigen Randes liegen → nur Support-Vektoren beeinflussen die Klassifizierung neuer Datenpunkte
- Sei S die Menge der Support-Vektoren und $\alpha_s \in \mathbb{R}$, $s \in S$, dazugehörige Parameter. Der Klassifizierer lässt sich in folgender Form schreiben:

$$f(x) = w_0 + \sum_{s \in S} \alpha_s \, (x \cdot x_s)$$

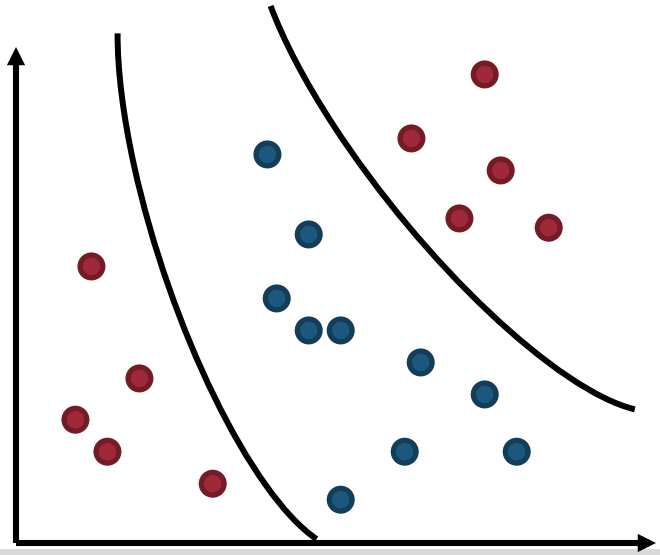
Support Vector Machine

- Erweiterung des (linearen) Support Vector Classifier über *Kernel:*

$$f(x) = w_0 + \sum_{s \in S} \alpha_s \, K(x, x_s)$$

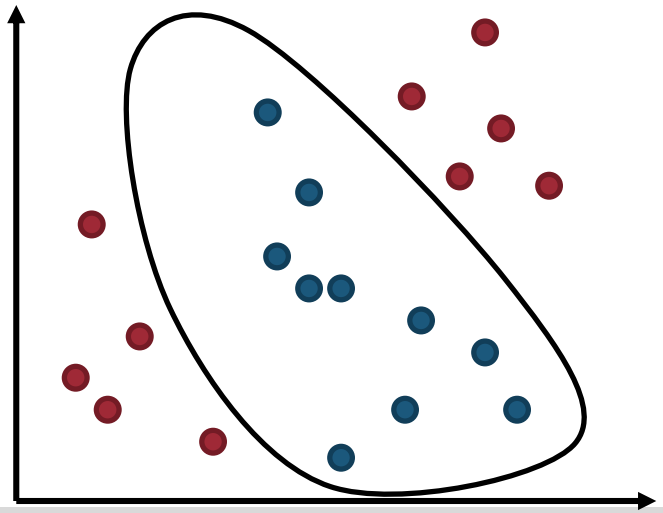
Polynomial Kernel

$$K(x, x_s) = (1 + x \cdot x_s)^d$$



Radial Kernel

$$K(x, x_s) = \exp\left(-r \sum_{i=1}^{n} (x_i - x_{si})^2\right)$$



Vergleich

Support Vector Machines	Entscheidungsbäume
Klassifikation: gewöhnlich 2 Klassen (erweiterbar auf mehrere Klassen)	Regression & Klassifikation (mehrere (~10) Klassen)
Attribute sind numerisch bzw. binär	Attribute sind numerisch oder kategorial
Verwendung tausender, spärlich besetzter Attribute	Beschränkung auf wenige, dicht besetzte Attribute
Schwer interpretierbar	Gut interpretierbar
Beispiele: • Textklassifikation (Genre, Sentiment Analysis, Spamfilter) • Gesichtserkennung	Beispiele: • Kundenklassifikation (Segmentierung) • Operations Research (Entscheidungen in Unternehmen)

Inhaltsverzeichnis

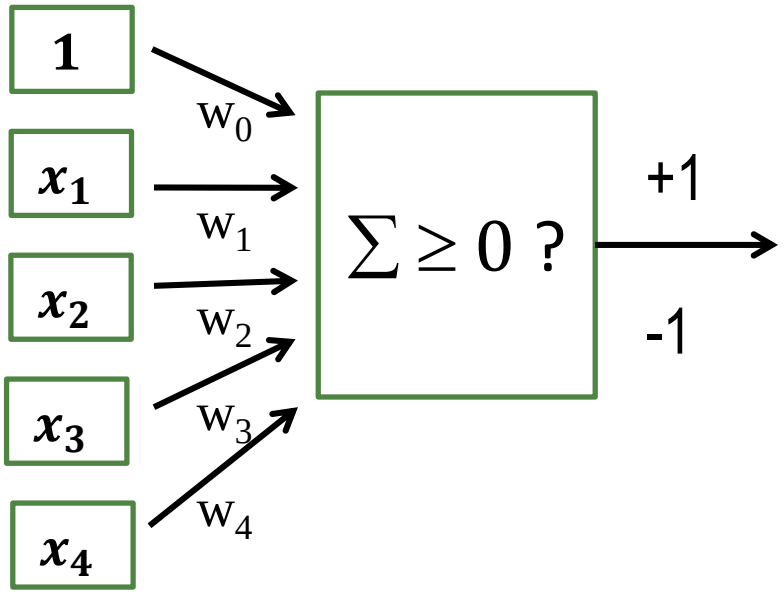
Literatur: Kapitel 12 und 13 aus „Mining of Massive Datasets“ (**v3.0 beta**): http://www.mmds.org

Lineare Klassifizierung

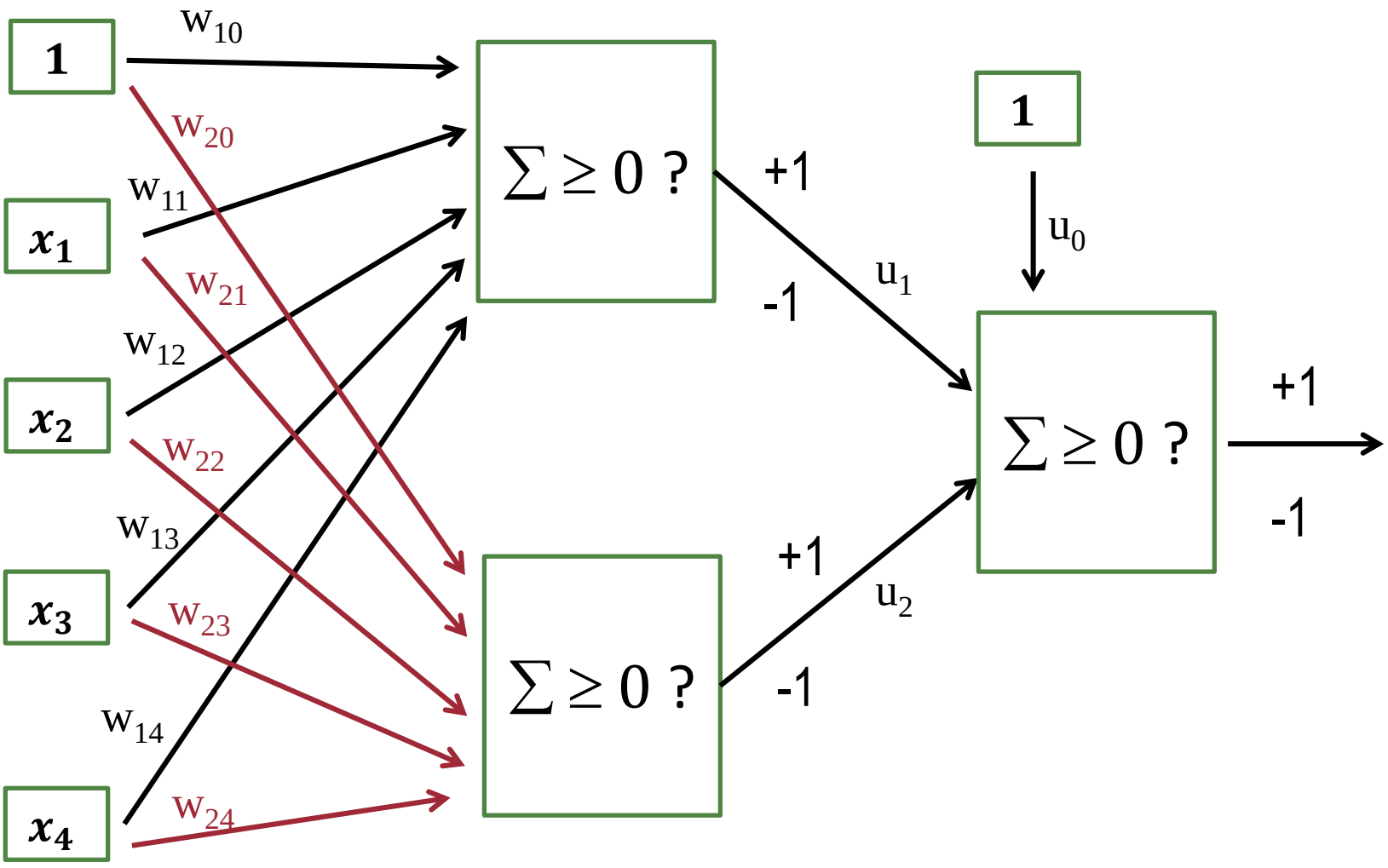
Klassifizierung über Hyperebene:

$$\hat{y} = +1, \text{ falls } w_0 + \sum_i w_i x_i > 0$$

$$\hat{y} = -1, \text{ falls } w_0 + \sum_i w_i x_i < 0$$



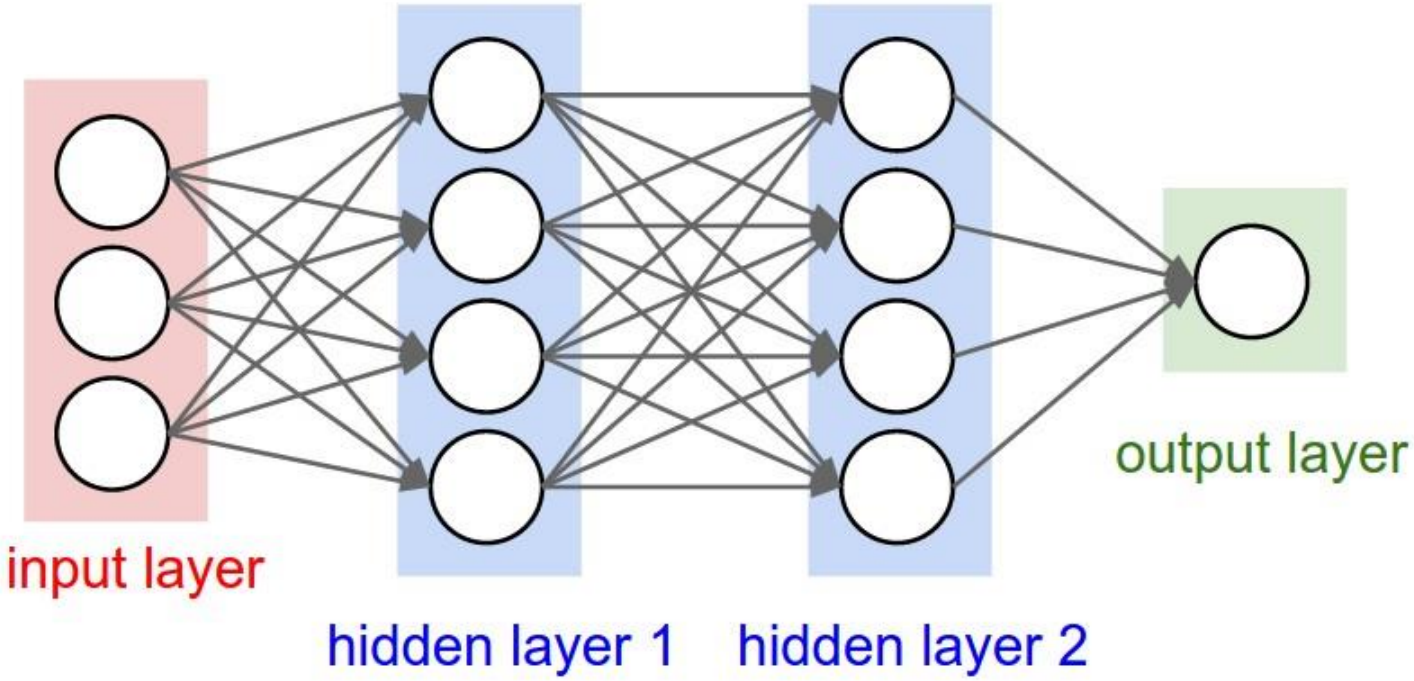
Erweiterte lineare Klassifizierung



Neuronale Netze

Neuronale Netze bestehen aus, in Schichten angeordnete, Knoten, wobei die Ausgabe einer Schicht die Eingabe der nächsten Schicht darstellt

- Eingabeschicht
- Mehrere versteckte Schichten
- Ausgabeschicht

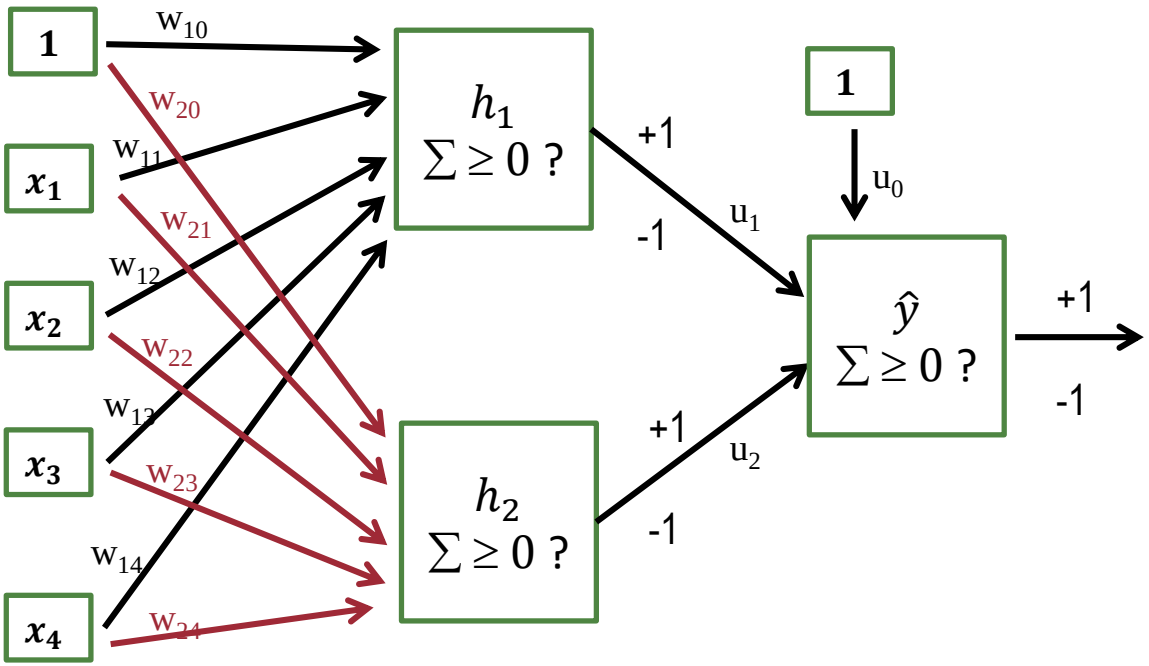


Quelle: http://cs231n.github.io/neural-networks-1/

Entwurf neuronaler Netze

1. Wie viele versteckte Schichten?
2. Wie viele Knoten pro Schicht?
3. **Berechnungen in den Knoten?**
4. **Kostenfunktion zur Bewertung der Ergebnisse?**
5. **Optimierungsalgorithmus zum Lernen der Gewichte?**
6. **Wie sind die Knoten einer Schicht zu den Knoten der nächsten Schicht verbunden?**

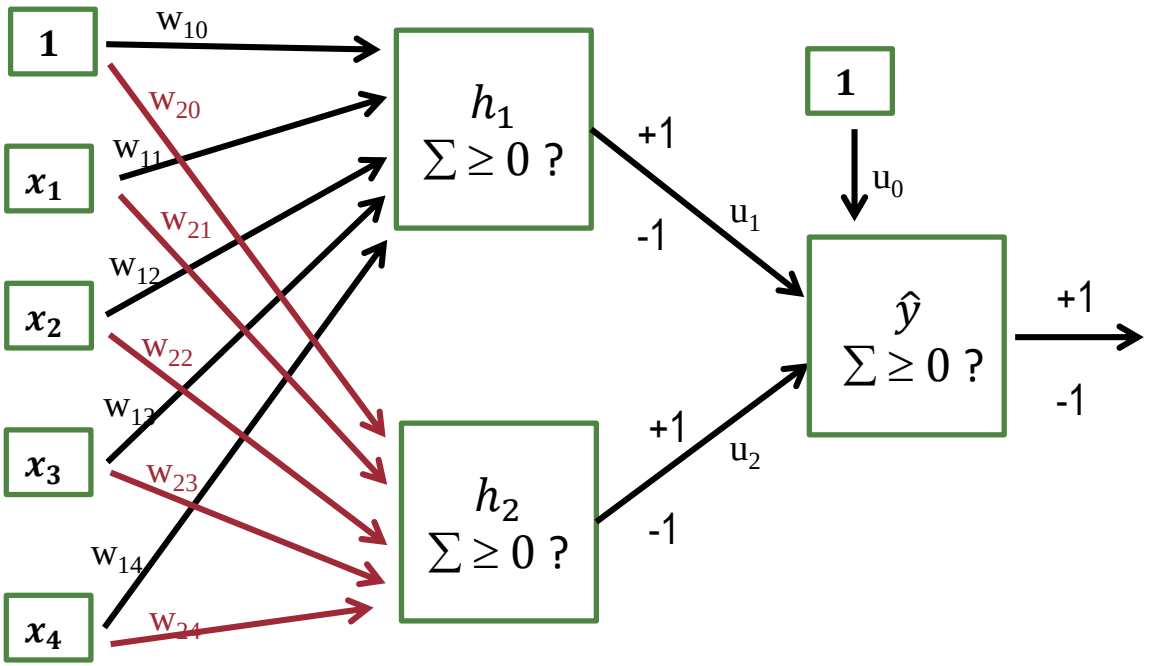
Berechnungen in den Knoten



Seien $x = \begin{pmatrix} x_1 \\ \dots \\ x_4 \end{pmatrix}, w_i = \begin{pmatrix} w_{i1} \\ \dots \\ w_{i4} \end{pmatrix}$ für $i = 1,2, u = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}$ und

$$step(z) = \begin{cases} 1 & , \text{falls } z \geq 0 \\ -1 & , \text{sonst} \end{cases}$$

Berechnungen in den Knoten



Dann gilt:

$$h_i = step\left(w_i^T x + w_{i0}\right) \text{ für } i = 1,2 \text{ und,}$$

$$\text{mit } h = \begin{pmatrix} h_1 \\ h_2 \end{pmatrix},$$

$$\hat{y} = step(u^T h + u_0)$$

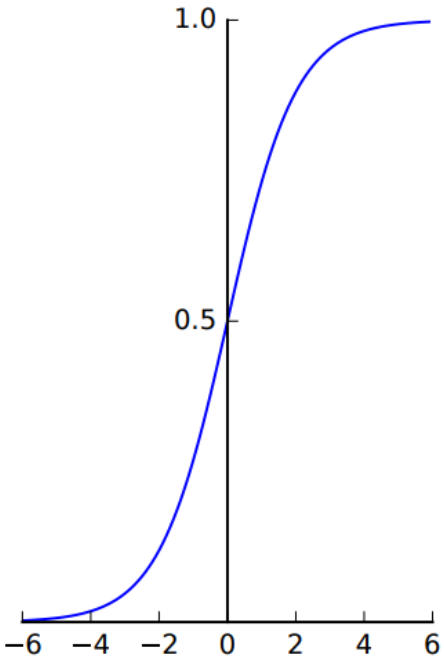
Aktivierungsfunktion

- Lernen der Gewichte über **Gradient Descent**
- Gute Funktionen für Gradient Descent:
  1. Stetig und (fast überall) differenzierbar
  2. Ableitung wird nicht zu klein (über dem erwarteten Wertebereich)
  3. Ableitung wird nicht zu groß (über dem erwarteten Wertebereich)
- Funktion $step(z)$ erfüllt die letzten beiden Kriterien nicht
- Alternativen
  - Sigmoid
  - Hyperbolic Tangent
  - Rectified Linear Unit
  - Exponential Linear Unit

Aktivierungsfunktionen

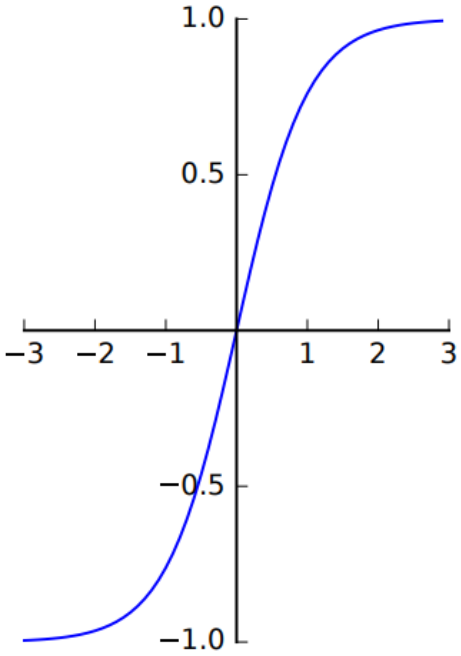
Sigmoid

$$\sigma(x) = \frac{e^x}{1 + e^x}$$



Hyperbolic Tangent

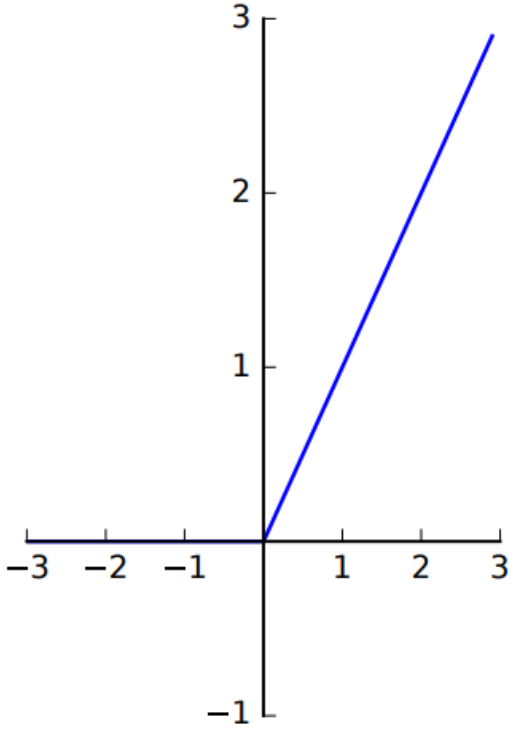
$$tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$



Aktivierungsfunktionen

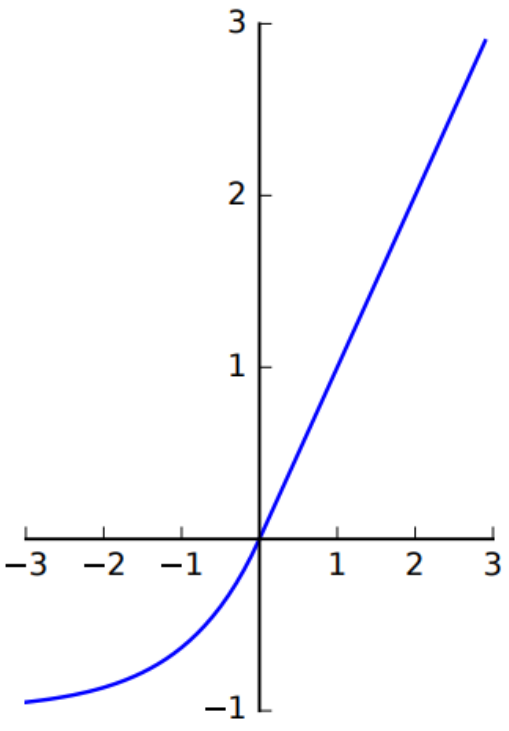
ReLU (Rectified Linear Unit)

$$f(x) = \max(0, x)$$



ELU (Exponential Linear Unit)

$$f(x) = \begin{cases} x & , x \geq 0 \\ \alpha(e^x - 1) & , x < 0 \end{cases}$$



Kostenfunktion zur Bewertung der Ergebnisse

- Variable Y: numerisch (Regression)
- Sei N die Anzahl der Datenpunkte, y_j die tatsächlichen Bezeichnungen und $\hat{y}_j$ die, durch das Neuronale Netz vorhergesagten, Bezeichnungen
- **Mean Squared Error:**

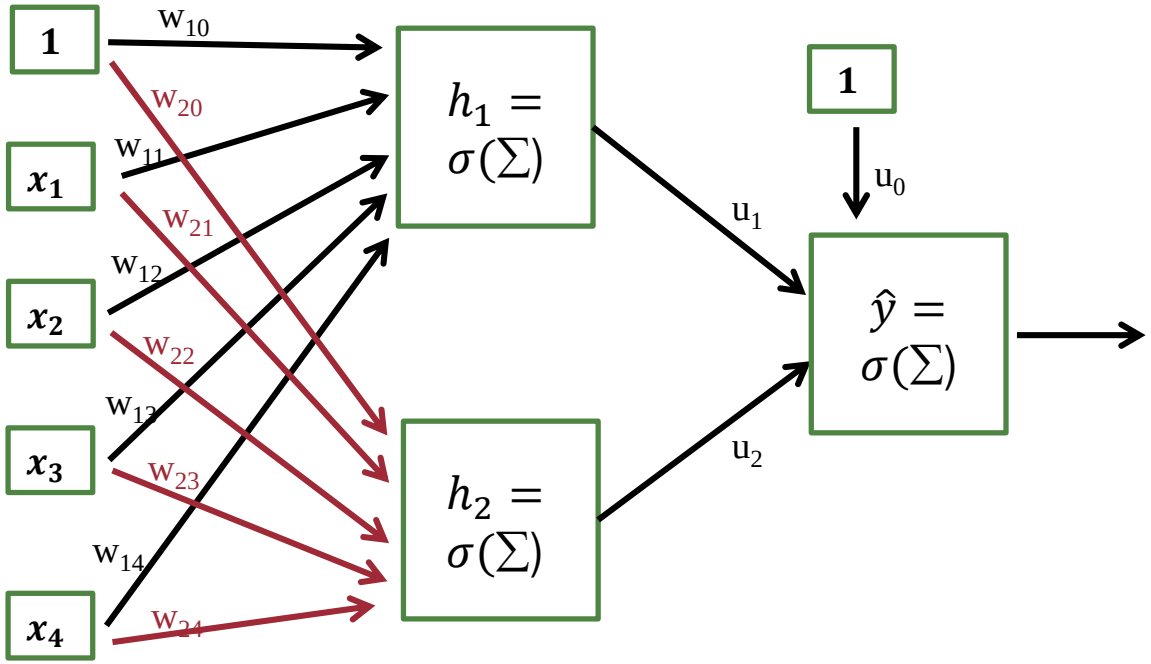
$$MSE = \frac{1}{N}\sum_{j=1}^{N}\left(y_j - \hat{y}_j\right)^2$$

- *Gradient*

$$\nabla_{\hat{y}_j} MSE = \frac{\partial MSE}{\partial \hat{y}_j} = \frac{1}{N} 2\left(\hat{y}_j - y_j\right)$$

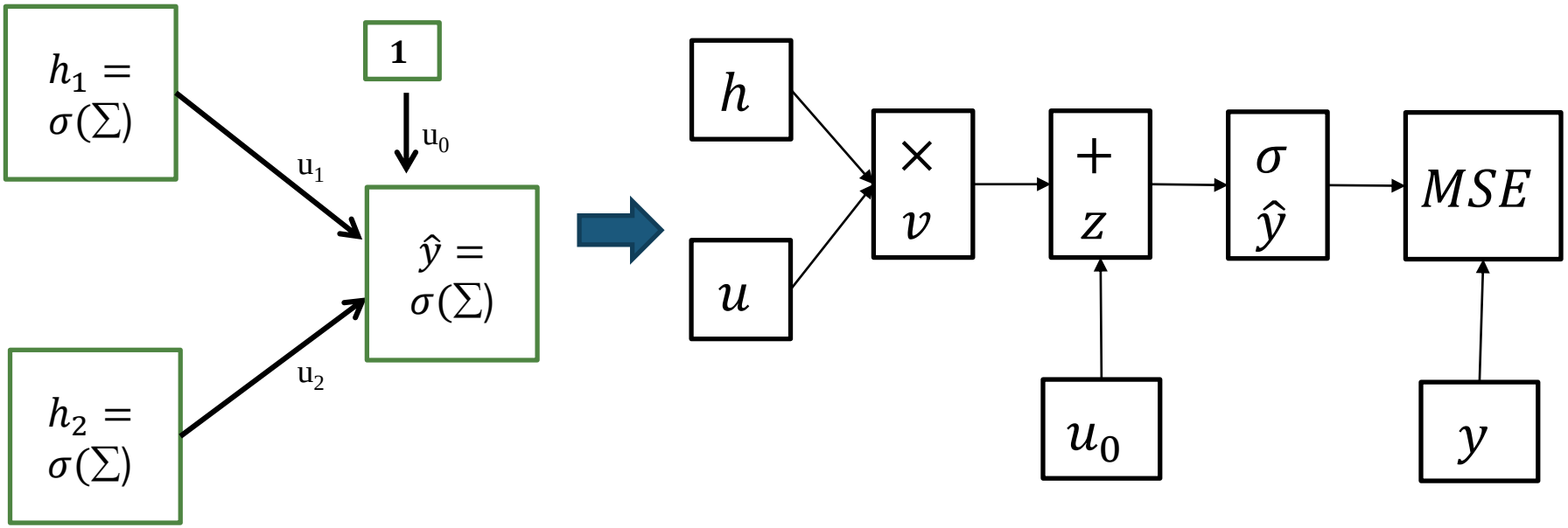
Optimierungsalgorithmus zum Lernen der Gewichte

- Anpassung der Gewichte (w_{ij} und u_j) mit dem Ziel der Minimierung der Kostenfunktion
- Effizientes Verfahren: **Gradient Descent** mit **Backpropagation**



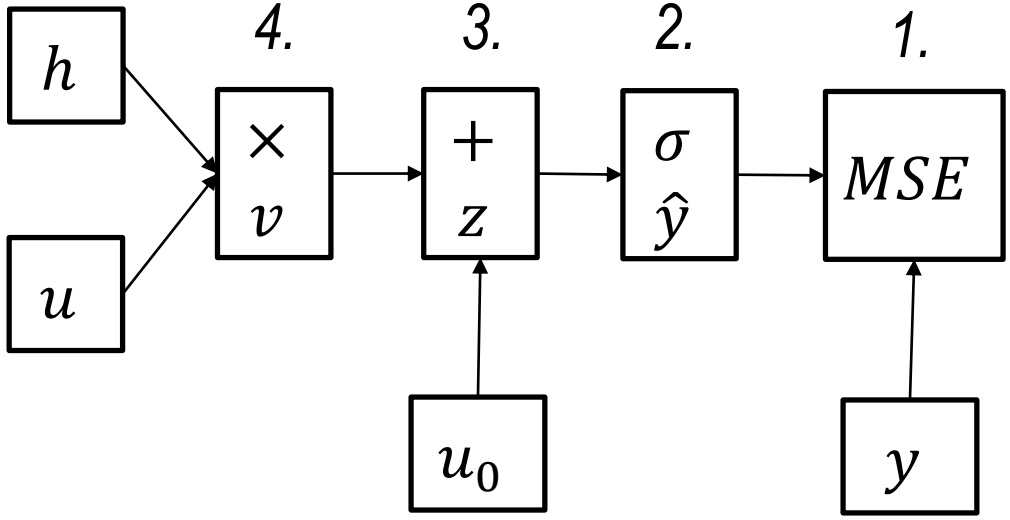
Berechnungsgraph

- Darstellung der Teilberechnungen eines (Abschnittes des) Neuronalen Netzes
- Einfügen von Variablen für Teilberechnungen
- Lernen der Gewichte über entgegengesetzte Richtung des Berechnungsgraphen (von hinten nach vorne)



Gradienten für jeden Teilschritt

- Ein Datenpunkt $((h_1, h_2)^T, y)$
- Gradienten für $u = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}$ und u_0

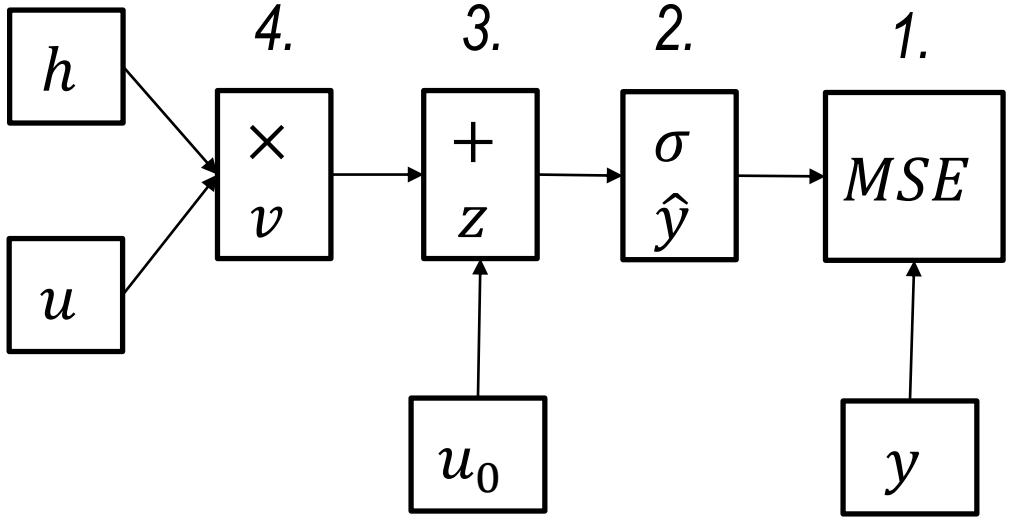


1. $\nabla_{\hat{y}} MSE = 2(\hat{y} - y)$
2. $\nabla_z \hat{y} = \nabla_z \sigma(z) = \hat{y}(1 - \hat{y})$
3. $\nabla_{u_0} z = \nabla_{u_0}(v + u_0) = 1$ und $\nabla_v z = \nabla_v (v + u_0) = 1$
4. $\nabla_u v = \nabla_u (u^T h) = \left[\frac{\partial v}{\partial u_1}, \frac{\partial v}{\partial u_2}\right]^T = \begin{pmatrix} h_1 \\ h_2 \end{pmatrix}$ und

   $\nabla_h v = \nabla_h (u^T h) = \left[\frac{\partial v}{\partial h_1}, \frac{\partial v}{\partial h_2}\right]^T = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}$

Backpropagation

- Ein Datenpunkt $\left((h_1, h_2)^T, y\right)$
- Gradienten für $u = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}$ und u_0

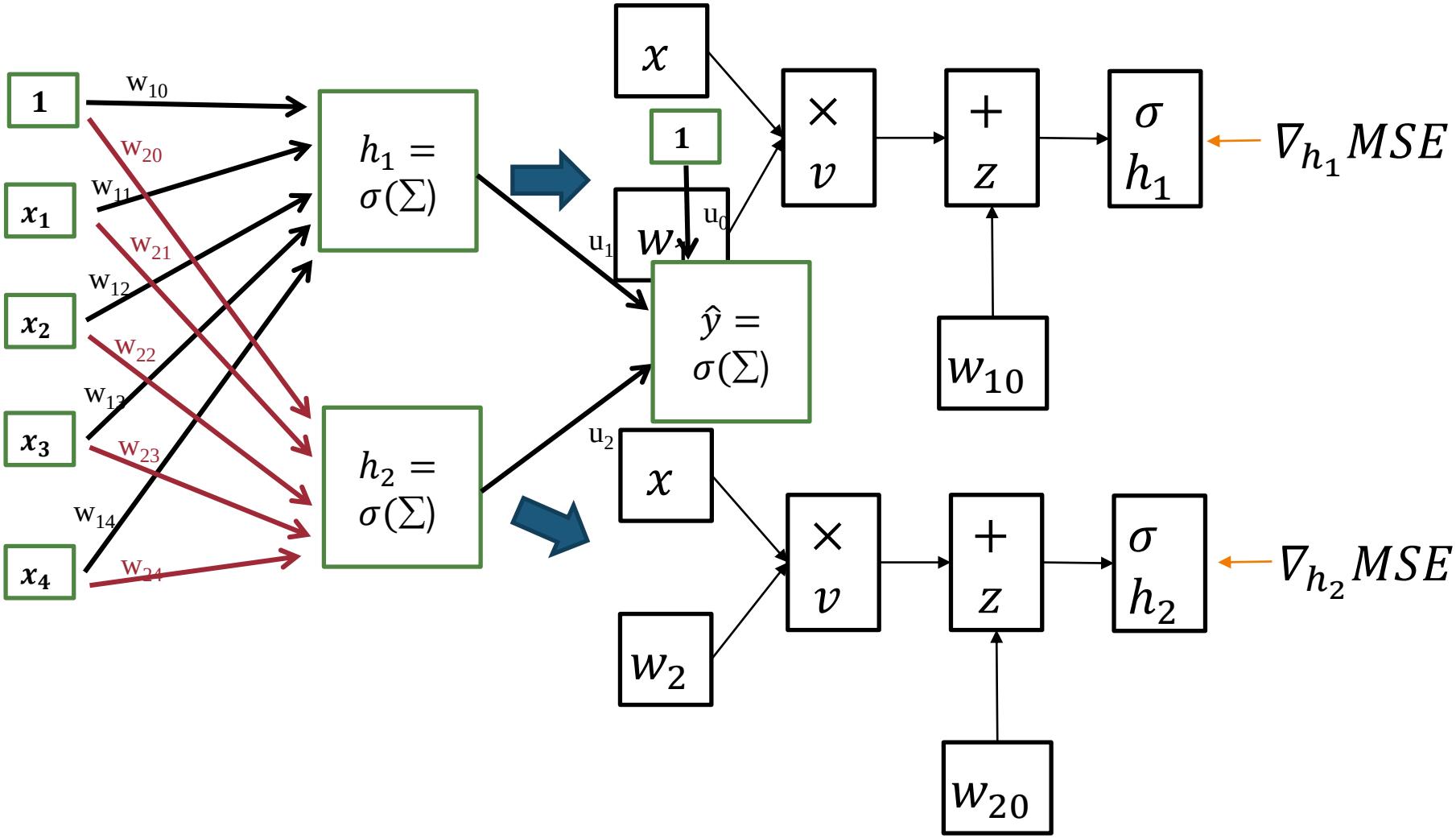


1. $\nabla_{\hat{y}} MSE = 2(\hat{y} - y)$
2. $\nabla_z MSE = \nabla_z \hat{y} \, \nabla_{\hat{y}} MSE = \hat{y}(1 - \hat{y}) 2(\hat{y} - y)$
3. $\boldsymbol{\nabla_{u_0} MSE} = \nabla_{u_0} z \, \nabla_z MSE = \hat{y}(1 - \hat{y}) 2(\hat{y} - y)$
4. $\boldsymbol{\nabla_u MSE} = \nabla_u v \, \nabla_v z \nabla_z MSE = \begin{pmatrix} h_1 \, \hat{y}(1 - \hat{y}) \, 2(\hat{y} - y) \\ h_2 \, \hat{y}(1 - \hat{y}) \, 2(\hat{y} - y) \end{pmatrix}$ und

$$\nabla_h MSE = \nabla_h v \, \nabla_v z \nabla_z MSE = \begin{pmatrix} u_1 \, \hat{y}(1 - \hat{y}) \, 2(\hat{y} - y) \\ u_2 \, \hat{y}(1 - \hat{y}) \, 2(\hat{y} - y) \end{pmatrix}$$

Backpropagation

Weiterreichen in nächste Schicht:



Gradient Descent

- Für N Datenpunkte mit $(x_1, y_1), \ldots, (x_N, y_N)$: arithmetisches Mittel der Gradienten: z.B.

$$\nabla_u MSE = \frac{1}{N} \sum_{j=1}^{N} \begin{pmatrix} h_{j1}\, \hat{y}_j(1-\hat{y}_j)\, 2\left(y_j - \hat{y}_j\right) \\ h_{j2}\, \hat{y}_j(1-\hat{y}_j)\, 2\left(y_j - \hat{y}_j\right) \end{pmatrix}$$

- Für jeden Parametervektor p (d.h. Gewichte $w_1, w_{10}, w_2, w_{20}, u, u_0$):

$$p \leftarrow p - \eta \nabla_p MSE$$

- Wiederholung bis Konvergenz

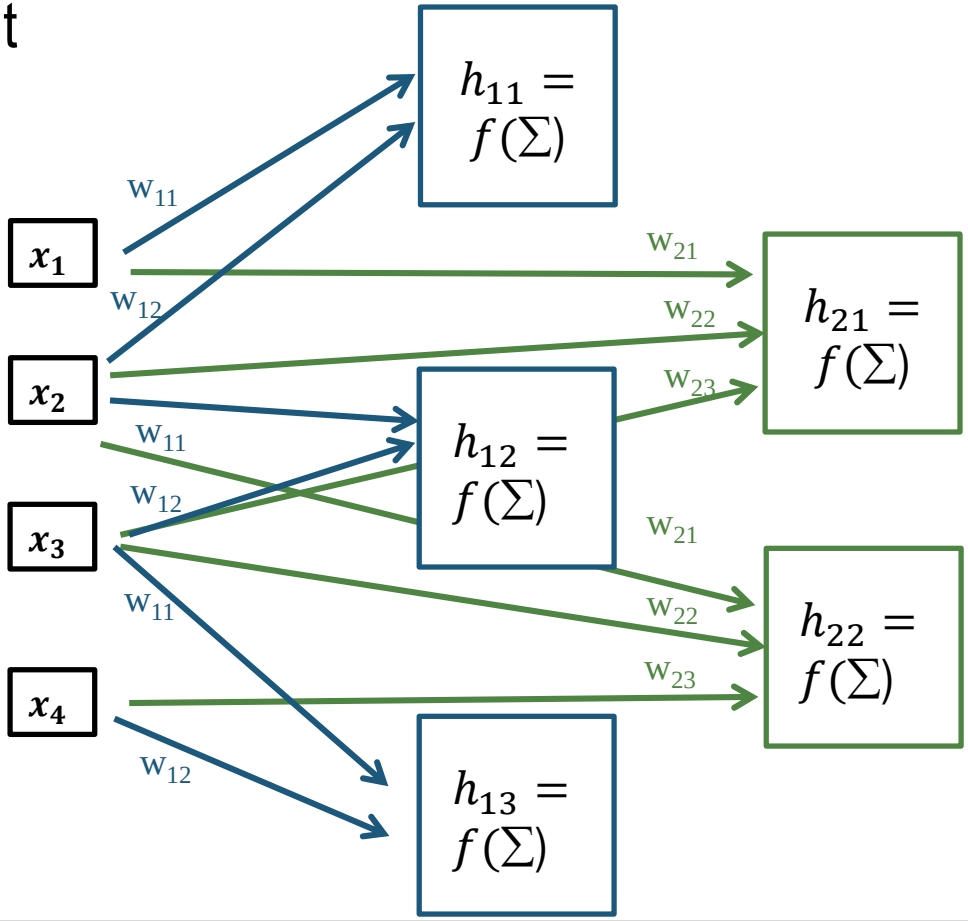
- *Stochastic* Gradient Descent für große Datensätze: Verwendung einer zufälligen Stichprobe aus den Datenpunkten pro Iteration

Entwurf neuronaler Netze

1. Wie viele versteckte Schichten?
2. Wie viele Knoten pro Schicht?
3. Berechnungen in den Knoten?
4. Kostenfunktion zur Bewertung der Ergebnisse?
5. Optimierungsalgorithmus zum Lernen der Gewichte?
6. **Wie sind die Knoten einer Schicht zu den Knoten der nächsten Schicht verbunden?**
   - Bisher: vollständig vernetzte Schichten (jeder Knoten ist verbunden zu allen Knoten der benachbarten Schichten)
   - Nachteil: Sehr viele Parameter bei großen Netzen oder hochdimensionalen Eingabedaten (z.B. Bilder/Texte)

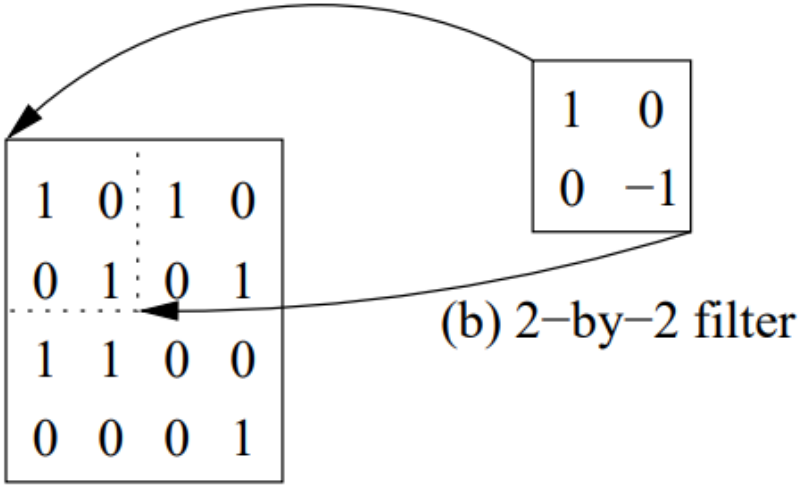
Convolutional Neural Networks (CNN)

- **Filter** = Einteilung einer Schicht in Partitionen und Zuordnung jeder Partition zu nur einem Knoten der folgenden Schicht
- Filter verwendet die selben Gewichte für jede Partition
- Mehrere Filter pro Schicht
- 1-dimensionale Eingabe und 2 Filter:



Convolutional Neural Networks (CNN)

- Filter für 2-dimensionale Eingabe:



- CNN eignen sich besonders für die Verarbeitung für Bildern
- Filter der ersten versteckten Schicht können darauf trainiert werden bestimmte einfache Formen zu erkennen, z.B. eine Ecke/Gerade
- Nachfolgende (Convolutional) Schichten können diese dann zu komplexeren Formen zusammensetzen, z.B. ein Kopf, eine Ampel,…

CNN: Beispiel

Erkennen handgeschriebener Zahlen

Quelle: https://en.wikipedia.org/wiki/MNIST_database

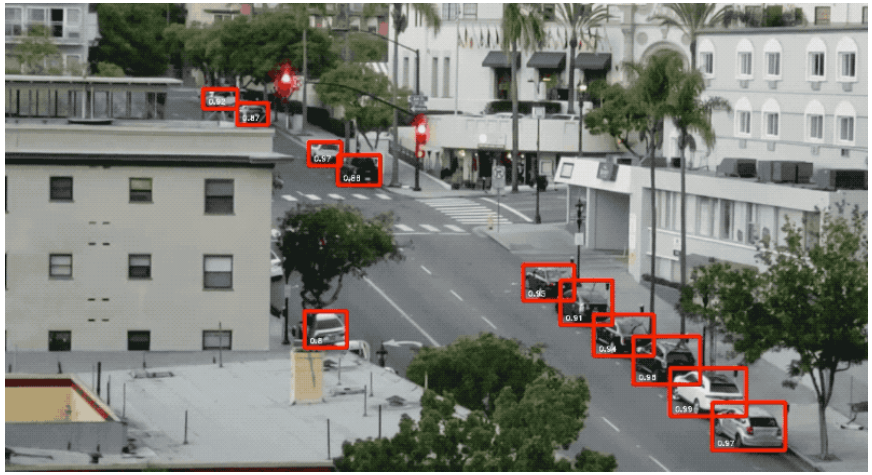
CNN: Beispiel

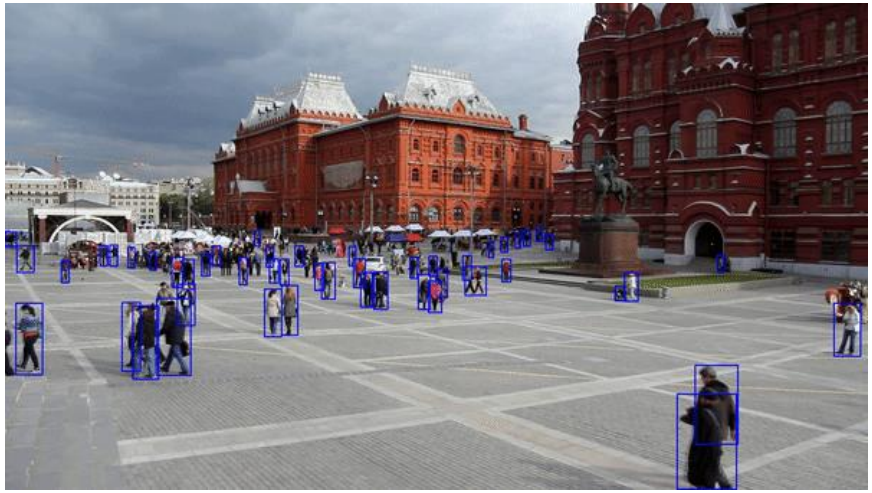
Erkennen von Kleidung (Bildsegmentierung)

Quelle: https://towardsdatascience.com/stuart-weitzman-boots-designer-bags-and-outfits-with-mask-r-cnn-92a267a02819

CNN: Beispiel

Erkennen von frei werdenden Parkplätzen / Zählung von Menschen



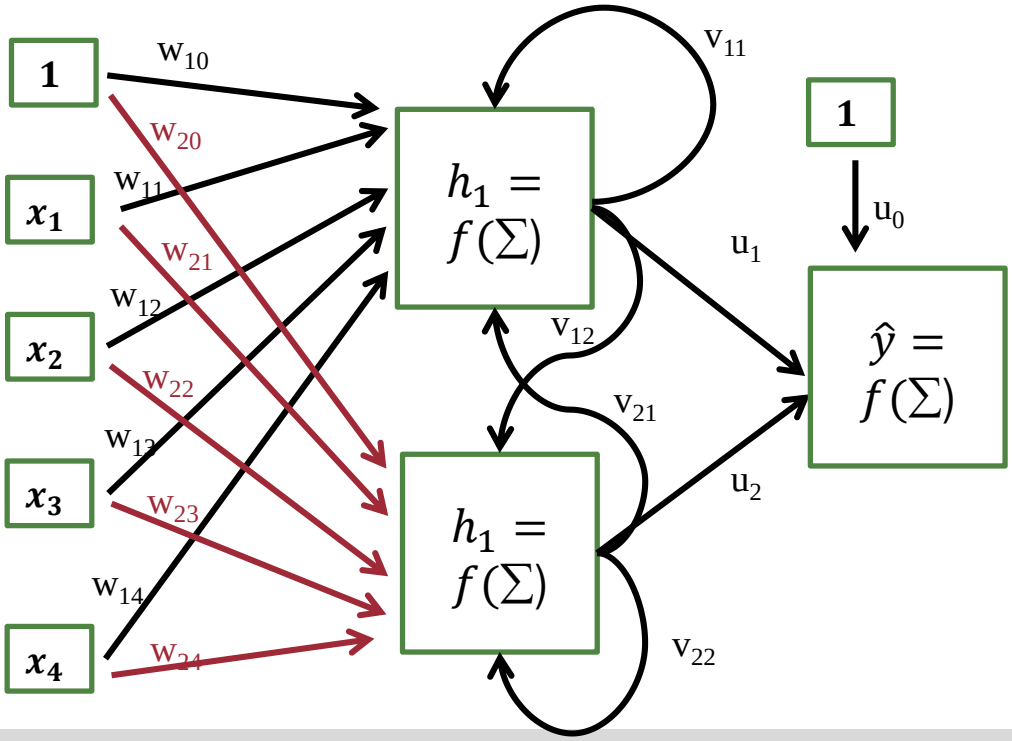


Quellen:

- https://medium.com/@ageitgey/snagging-parking-spaces-with-mask-r-cnn-and-python-955f2231c400
- https://nanonets.com/blog/crowd-counting-review/?utm_source=reddit&utm_medium=social&utm_campaign=drcrco&utm_content=dl

Recurrent Neural Networks (RNN)

- Spezielle Netzwerke für Verarbeitung von Sequenzen, z.B.
  - Text (Sequenz aus Wörtern)
  - Videos (Sequenz aus Bilder)
- Beispiel: Text
  - Eingabe $x = (x_1, \ldots, x_n)$ bezeichnet ein Wort (One-Hot Encoded: Dimension n ist die Anzahl der möglichen Wörter; genau eine Eins an der Stelle des Wortes, ansonsten nur Nullen)
  - Beachtung der Reihenfolge der Wörter über zyklische Verbindung innerhalb einer Schicht
  - Zyklische Schicht kann sich Zustände „merken“ und beeinflusst die Verarbeitung des nächsten Wortes



RNN: Beispiel

- Klassifizierung von Texten
- Positive oder negative Rezensionen

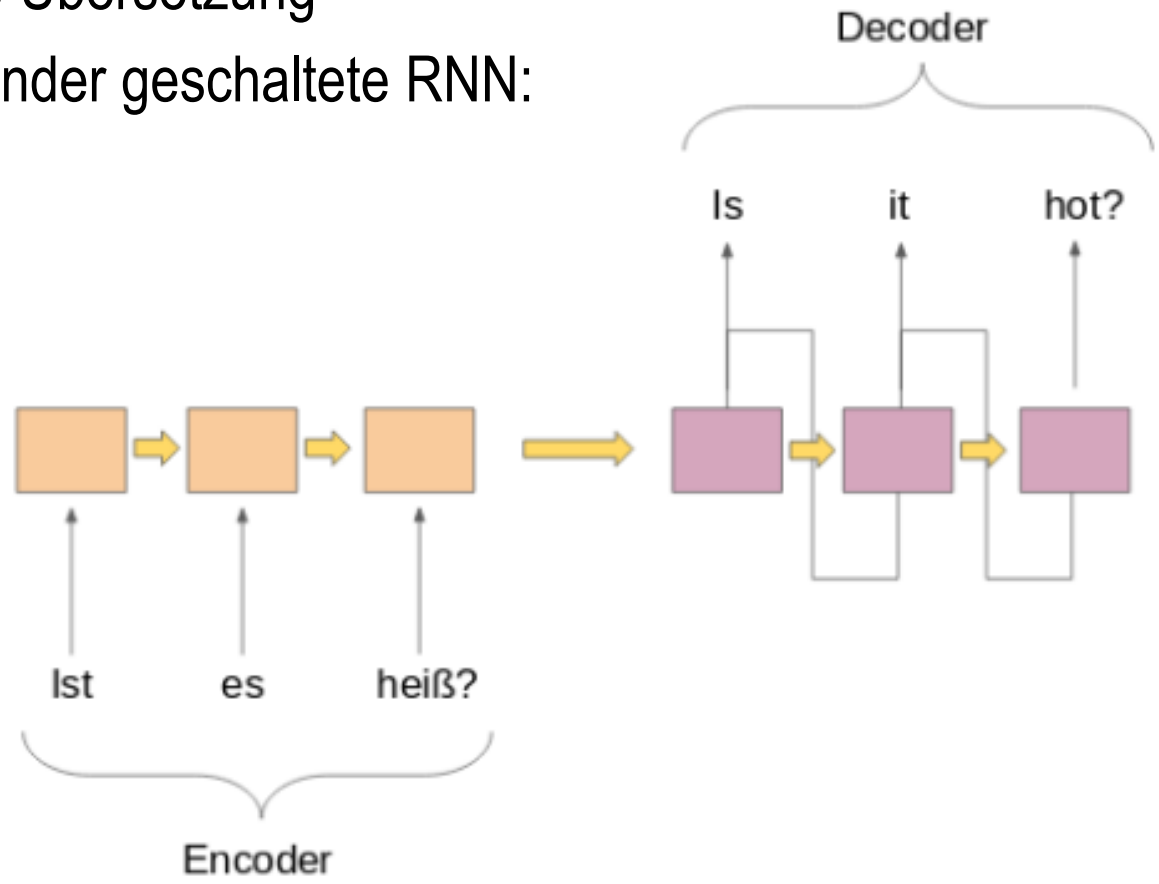
```
> sample_predict('The movie was cool. The animation and the graphics were out of this world. I would recommend this movie.')
[[0.4186573]]
```

```
> sample_predict('The movie was not good. The animation and the graphics were terrible. I would not recommend this movie.')
[[0.05746633]]
```

Quelle: https://www.tensorflow.org/tutorials/text/text_classification_rnn

RNN: Beispiel

- Maschinelle Übersetzung
- 2 hintereinander geschaltete RNN:



Quelle: https://medium.com/analytics-vidhya/a-must-read-nlp-tutorial-on-neural-machine-translation-the-technique-powering-google-translate-c5c8d97d7587

RNN: Beispiel

- Textgenerierung: z.B. Gedichte
- Eingabe des Nutzers: Thema oder Bild

Mountains

A hundred thousand Morrison formation,

An ancient crown of gold or mountain chains,

Mountains from the land of elevation,

A northern storm across the hills and plains.

the sun is a beautiful thing

in silence is drawn

between the trees

only the beginning of light

Quellen:

- http://xingshi.me/data/pdf/ACL2017demo.pdf
- https://arxiv.org/pdf/1804.08473.pdf

Inhaltsverzeichnis

Literatur: Kapitel 12 und 13 aus „Mining of Massive Datasets“ (**v3.0 beta**): http://www.mmds.org

Übung

Gegeben sind 7 Datenpunkte.

Ziel ist die Vorhersage von Y über X_1 und X_2.

a) Zeichnen Sie einen Entscheidungsbaum minimaler Größer, welcher alle Datenpunkte korrekt zuordnet!

b) Leiten Sie den Maximal Margin Classifier her!

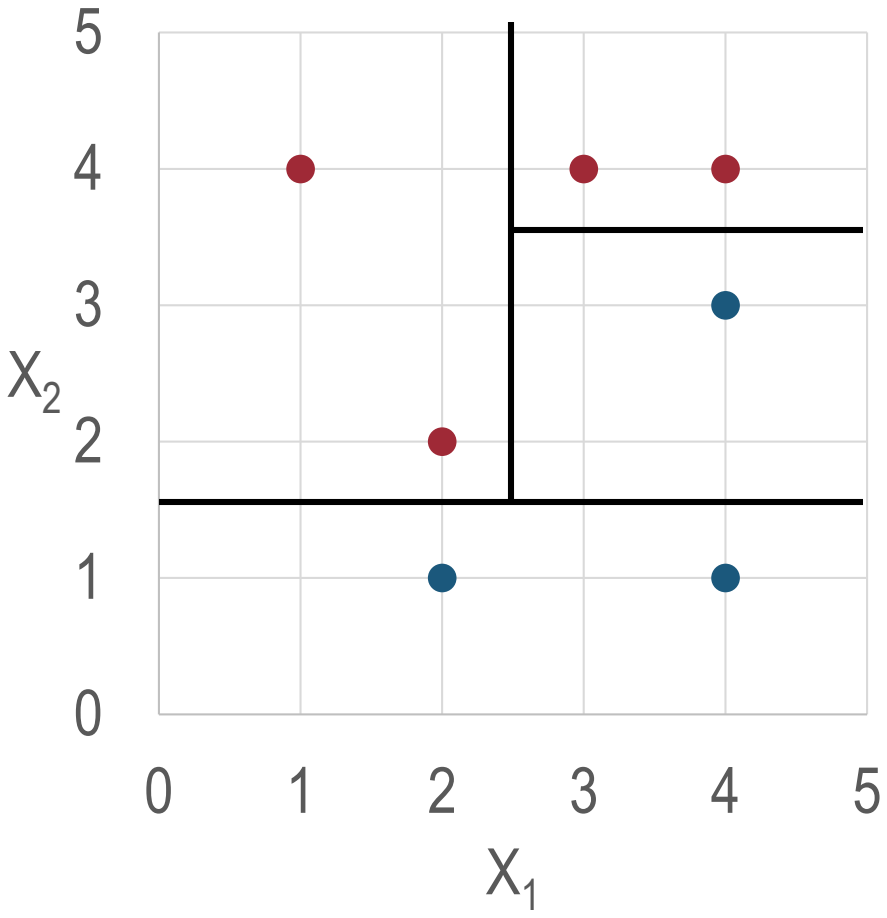
c) Welche Anpassungen wären in den beiden Modellen notwendig, wenn zusätzlich folgender Datenpunkt beobachtet wird:

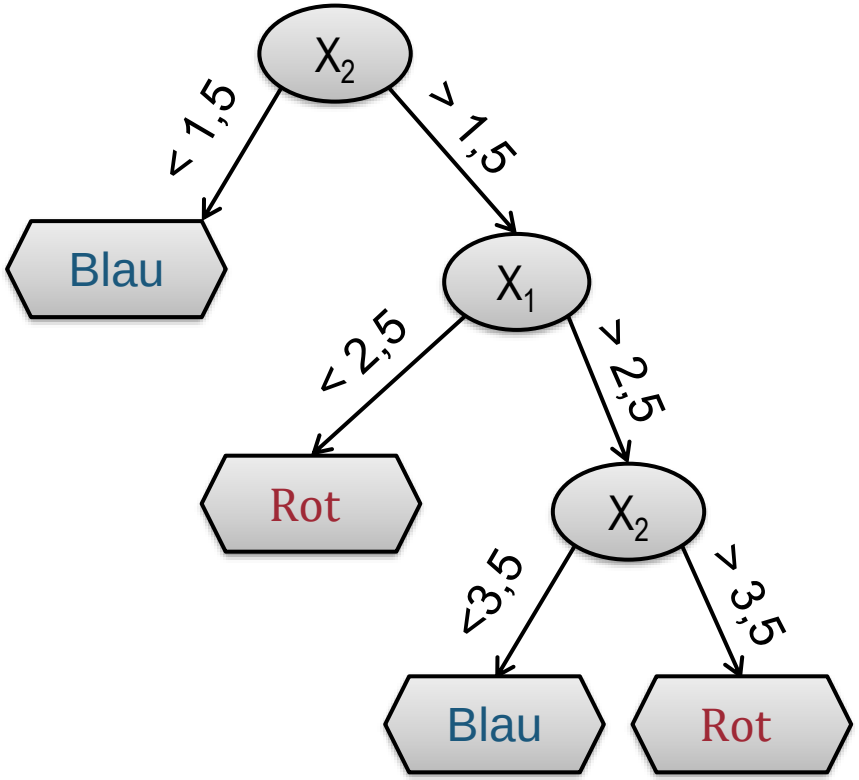
X_1	X_2	Y
3	3	Blau

X_1	X_2	Y
3	4	Rot
2	2	Rot
4	4	Rot
1	4	Rot
2	1	Blau
4	3	Blau
4	1	Blau

Übung: Lösung

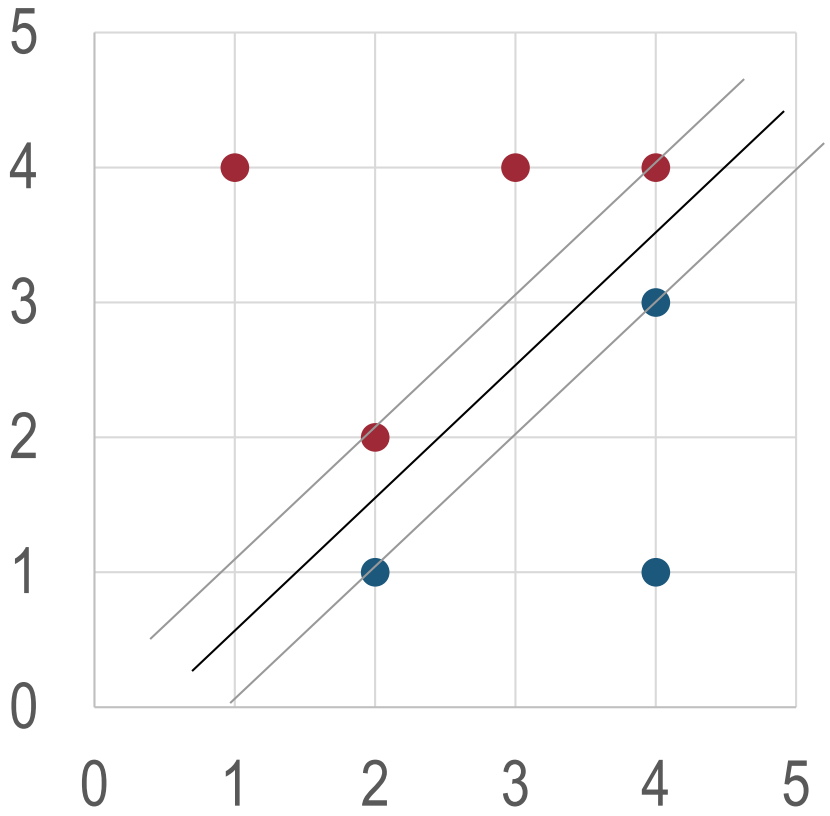
a) Zeichnen Sie einen Entscheidungsbaum minimaler Größer, welcher alle Datenpunkte korrekt zuordnet!





Übung: Lösung

b) Leiten Sie den Maximal Margin Classifier her!



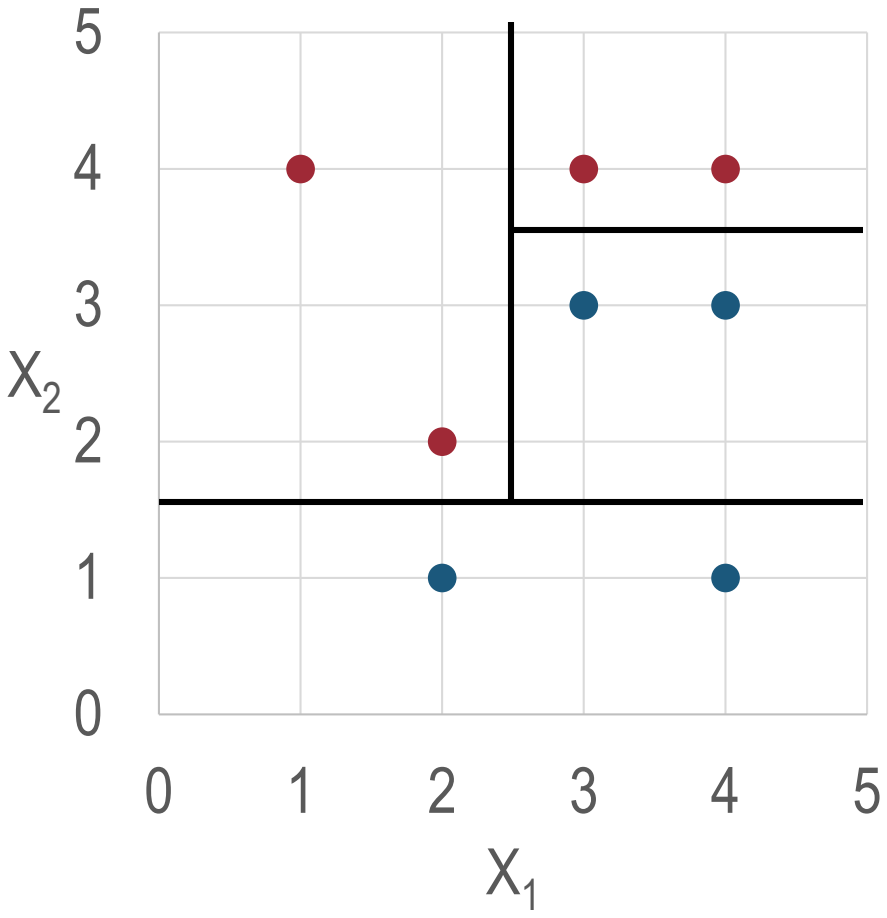
$$1 = w_0 + 4w_1 + 4w_2$$
$$1 = w_0 + 2w_1 + 2w_2$$
$$-1 = w_0 + 2w_1 + w_2$$
$$-1 = w_0 + 4w_1 + 3w_2$$

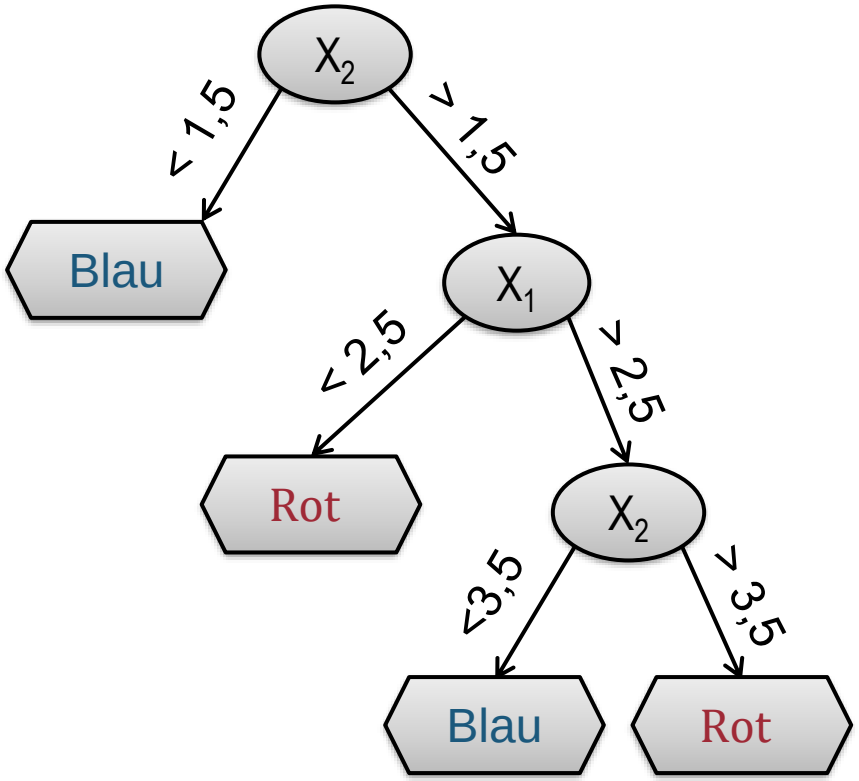
$$w_0 = 1$$
$$w_1 = -2$$
$$w_2 = 2$$

Übung: Lösung

c) Welche Anpassungen wären in den beiden Modellen notwendig, wenn zusätzlich folgender Datenpunkt beobachtet wird:

X_1	X_2	Y
3	3	Blau

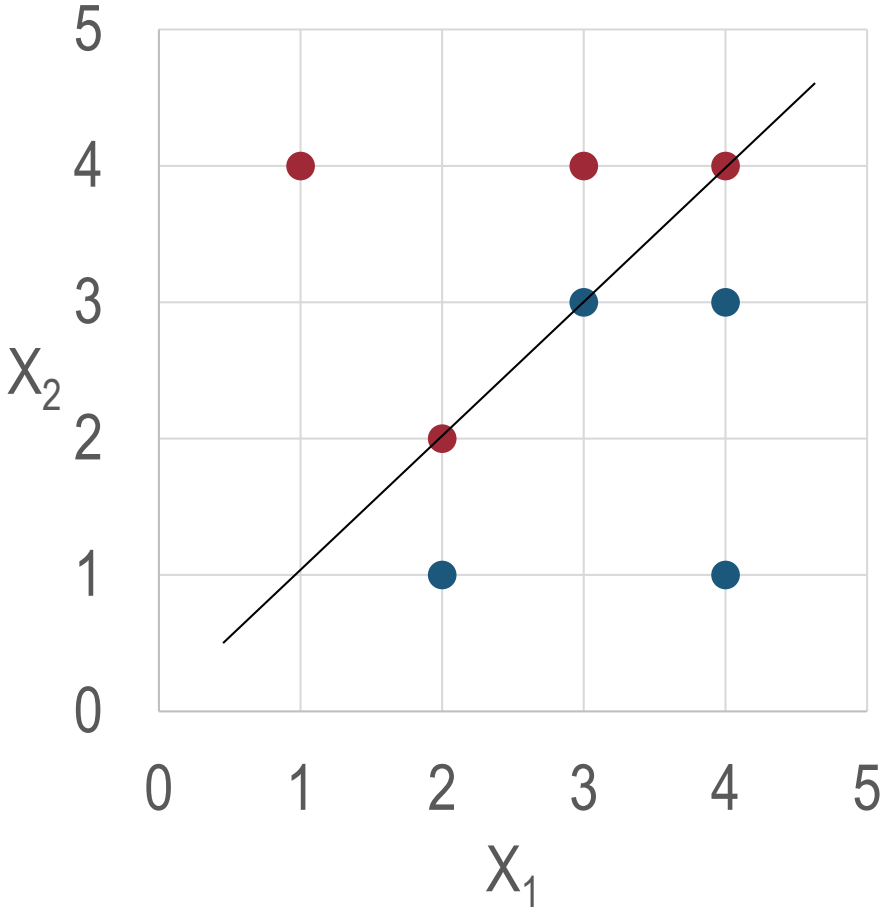




Übung: Lösung

c) Welche Anpassungen wären in den beiden Modellen notwendig, wenn zusätzlich folgender Datenpunkt beobachtet wird:

X_1	X_2	Y
3	3	Blau



$$1 = w_0 + 4w_1 + 4w_2$$
$$1 = w_0 + 2w_1 + 2w_2$$
$$-1 = w_0 + 3w_1 + 3w_2$$

Nicht linear trennbar!