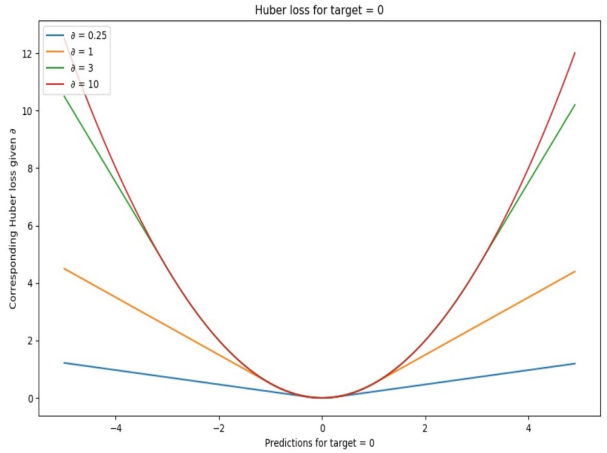
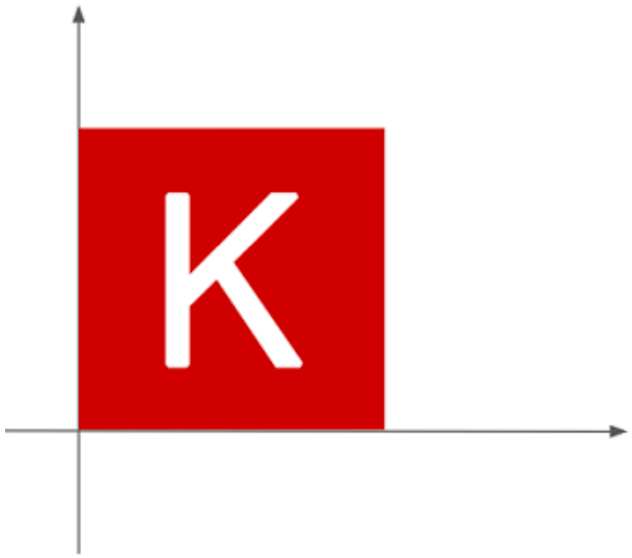
Recap von Tag 2

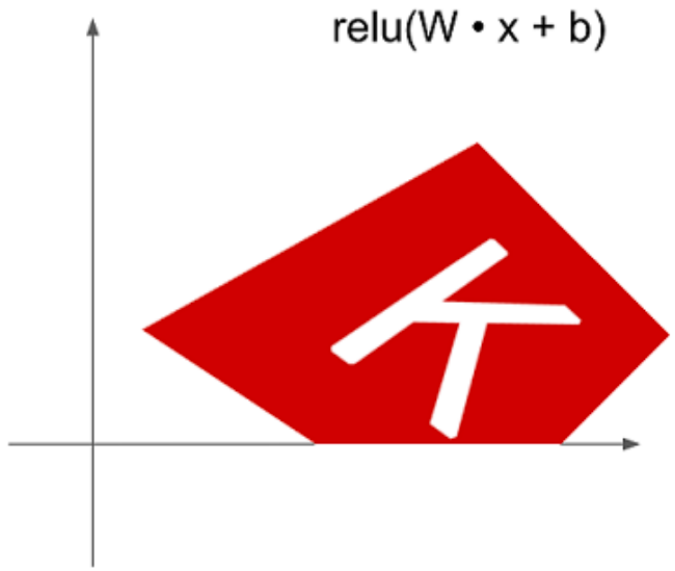
Was haben wir gestern gemacht?

Allgemeine Konzepte

- Unterschied zwischen Regressionen und Klassifikationen
- Loss-Funktionen
- Overfitting/Underfitting
- Geometrische Interpretation von Deep Learning







Bag-of-Word Modelle

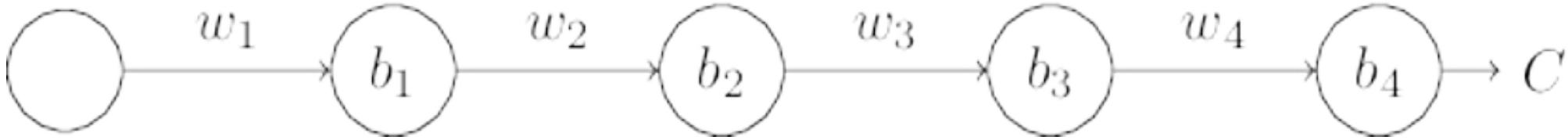
- Kodierung von Text-Datensätzen (Tokenization)
- Normalisierung von Trainingsdaten
- Visualisierung von Daten/Training-History
- K-Fold Validation
- Diagnose anhand von Validationloss und –accuracy
- Binäre Klassifikation und Multi-Klassifikation
- Einfluss von Layeranzahl, Parameteranzahl, Informations-Engpässen, Aktivationsfunktionen, etc.

Optimierung des Trainings neuronaler Netzwerke

Vorhersage von kontinuierlichen oder diskreten Werten

- Machine Learning Probleme unterscheiden sich von einem klassischen Optimierungsproblem
- Typischerweise ist das Ziel ein Performance Maß auf dem Testdatensatz zu optimieren dazu reduzieren wir die Loss Funktion in der Hoffnung damit auch das Performance Maß zu verbessern
- Die ist anders bei reiner Optimierung, wo die Minimierung des Losses das Ziel darstellt.
- Insbesondere das Problem des Overfittens existiert für Optimierungsprobleme nicht

Vanishing / Exploding Gradient Problem



- Betrachte ein 4 Layer feedforward Netz mit einem Neuron pro Layer
- Output des Netzwerks: a_4
- Output der j-ten Layer: $a_j = \sigma(z_j), \text{ mit } z_j = w_j a_{j-1} + b_j$
- Backpropagation gibt für den Gradienten in der ersten Layer:

$$\frac{\partial C}{\partial b_1} = \sigma'(z_1) \cdot w_2 \cdot \sigma'(z_2) \cdot w_3 \cdot \sigma'(z_3) \cdot w_4 \cdot \sigma'(z_4) \cdot \frac{\partial C}{\partial a_4}$$

- Produkt mit vielen Faktoren -> instabiler Gradient, denn
  - Exponentieller Anstieg für Faktoren > 1
  - Exponentieller Abfall für Faktoren < 1

Tips und Tricks gegen instabile Gradienten

- Dropout
- Gute Initialisierung von Gewichten (He- bzw- Glorot-Initialisierung)
- Gradient clipping
- Batch Normalization
- Layer Normalization

Glorot-Initialisierung & Gradient Clipping

Initialisierung der Gewichte einer Layer mit N_{in} Inputs und N_{out} mit einer Gaußverteilung oder uniformen Verteilung via:

$$\text{Normalverteilung mit } (\mu, \sigma^2) = \left(0, \frac{1}{N_{\text{avg}}}\right)$$

$$\text{Uniforme Verteilung aus dem Intervall } \left(-\sqrt{\frac{3}{N_{\text{avg}}}}, \sqrt{\frac{3}{N_{\text{avg}}}}\right)$$

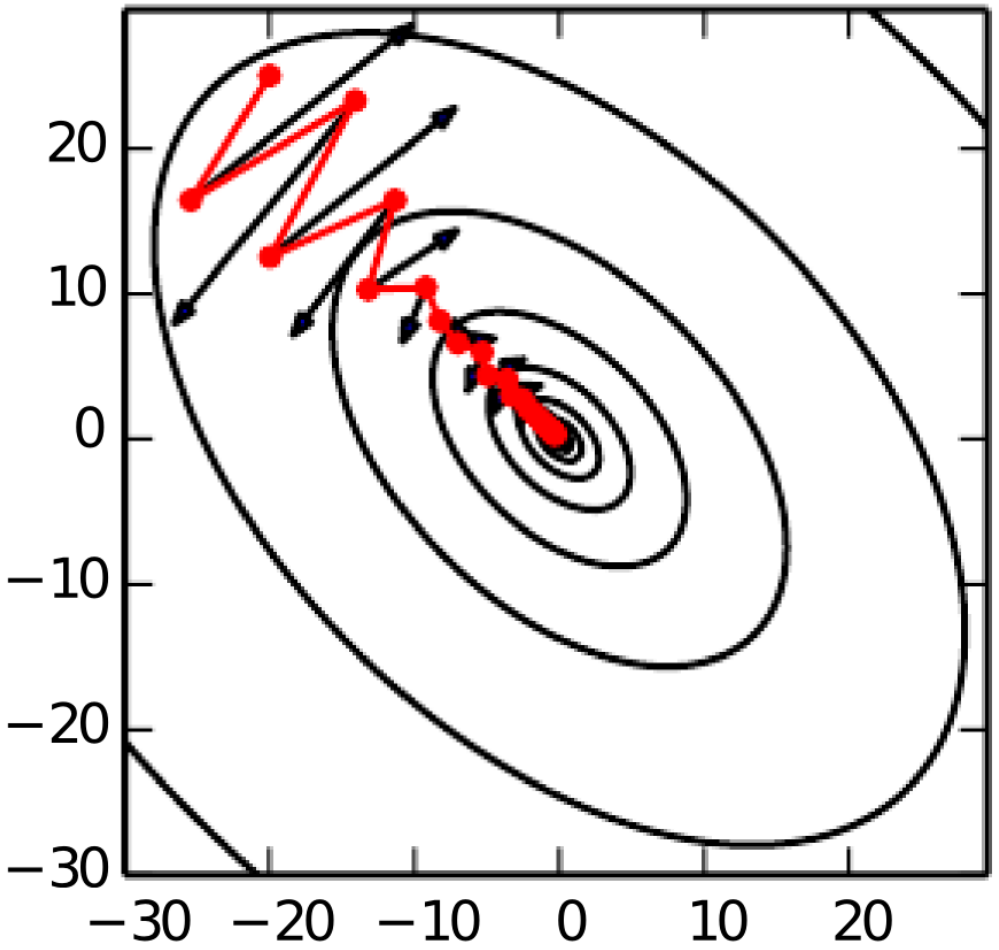
Gradient Clipping: Schneide den Gradienten ab

2 Möglichkeiten:

- Norm clipping: reduziert die Länge des Gradientenvektors auf einen bestimmten Wert (verändert die Richtung des Gradienten nicht)
- Clip value: reduziert die Komponenten des Gradientenvektors auf einen bestimmten Wert

Momentum

- **Führe Trägheitsmoment des Gradienten ein:** $\mathbf{v}_{\text{new}} = \alpha \mathbf{v}_{\text{old}} - \epsilon \mathbf{g}$
- **Exponentiell abfallend, da** $\alpha \in (0, 1)$
- **Update:** $\theta_{\text{new}} = \theta_{\text{old}} + \mathbf{v}_{\text{new}}$
- **Reduktion oszillierender Richtungen**
- **Beschleunigung in häufiger auftretenden Richtungen**



Nesterov's Accelerated Momentum

- **Berechne Gradienten nicht da wo er ist, sondern wo er sein wird**

$$\mathbf{g} = \frac{1}{n} \nabla_\theta \sum_{i=1}^{n} \mathcal{L}\left(f(\mathbf{x}^{(i)}; \theta + \alpha \mathbf{v}_{\text{old}}), \mathbf{y}^{(i)} \right)$$

- **Vorläufiger Parameterupdate** $\tilde{\theta} = \theta + \alpha \mathbf{v}_{\text{old}}$ **vor Gradientenberechnung**
- **Momentum:** $\mathbf{v}_{\text{new}} = \alpha \mathbf{v}_{\text{old}} - \epsilon \mathbf{g}$
- **Tatsächlicher Parameterupdate:** $\theta_{\text{new}} = \theta_{\text{old}} + \mathbf{v}_{\text{new}}$

AdaGrad

- **Optimizer für Gradientenabstieg**
- **Algorithmus:**

$$\mathbf{s}_{\text{new}} = \mathbf{s}_{\text{old}} + \mathbf{g} \otimes \mathbf{g},$$

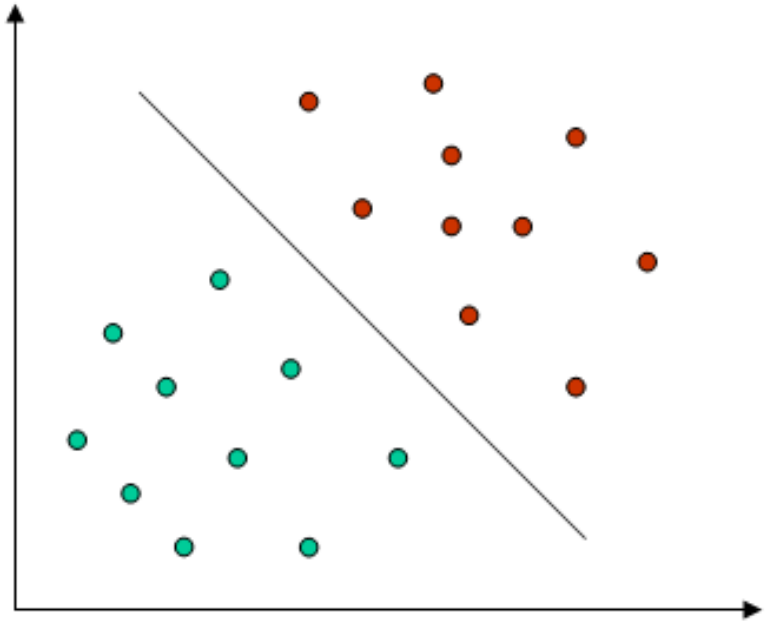
$$\theta_{\text{new}} = \theta_{\text{old}} - \epsilon \mathbf{g} \oslash \sqrt{\mathbf{s} + \eta}$$

- **Skaliere learning rate mit dem Quadrat der Momente**
- **D.h. in steilen Richtungen mache learning rate kleiner als in weniger steilen Richtungen**
- η **Ist ein Glättungsterm und verhindert underflow**
- **AdaGrad wird heute nicht mehr wirklich verwendet, bildet aber die Grundlage für das Verständnis moderner Optimizer (Adam, RMSProp)**

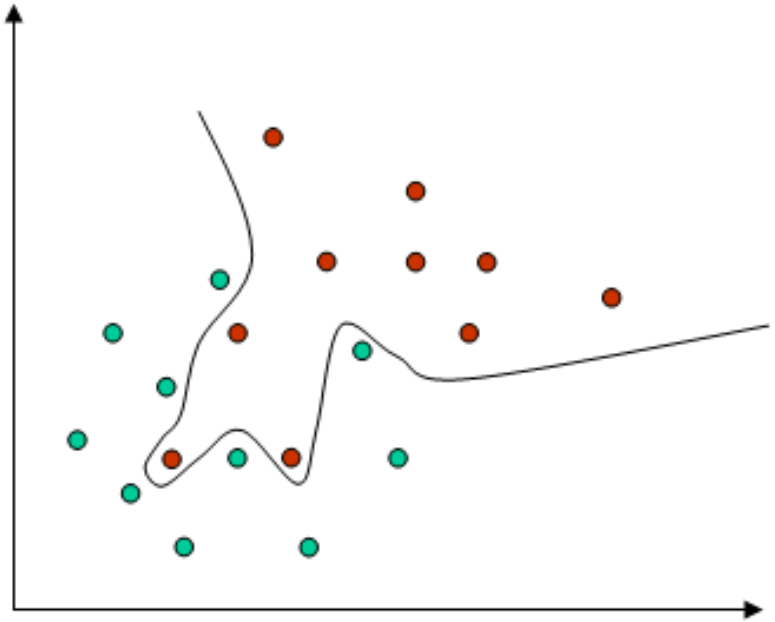
Die bekanntesten Deep Learning Forscher ….

Klassifikation – Support Vector Machine

- Erste grundlegende Idee: In einem höher-dimensionalem Raum werden Klassen linear trennbar (wird mit Hilfe des “Kernel tricks“ rechnerisch umgesetzt.)



linear trennbar

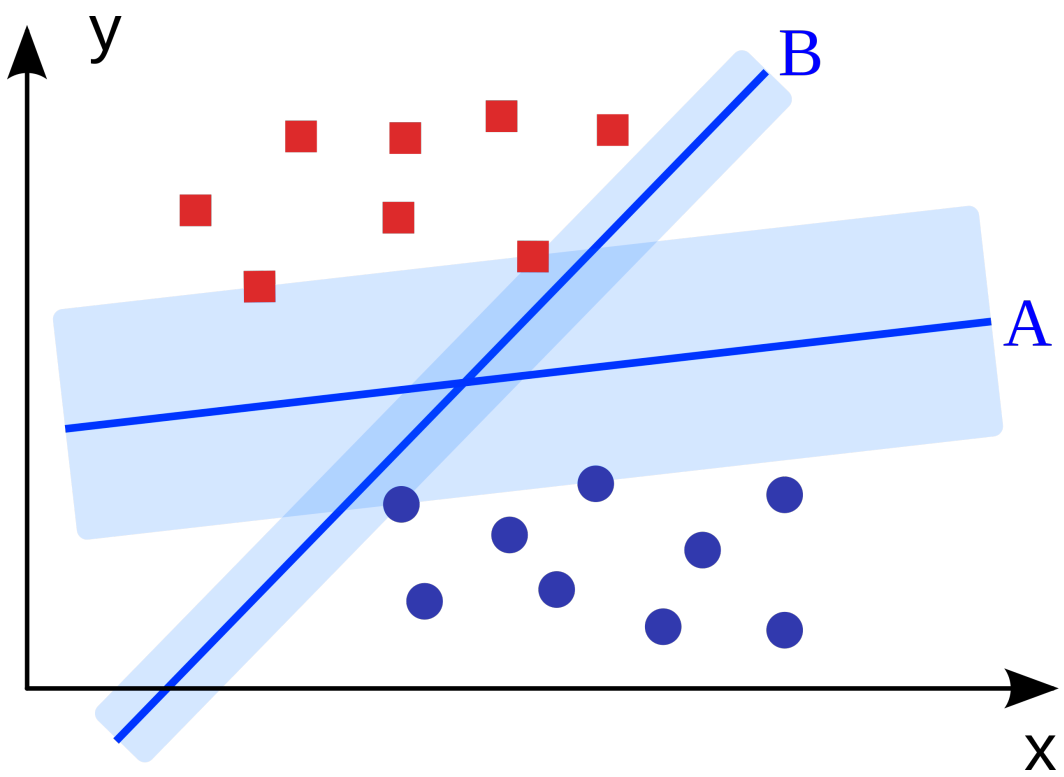


nicht linear trennbar

https://de.wikipedia.org/wiki/Support_Vector_Machine

Klassifikation – Support Vector Machine

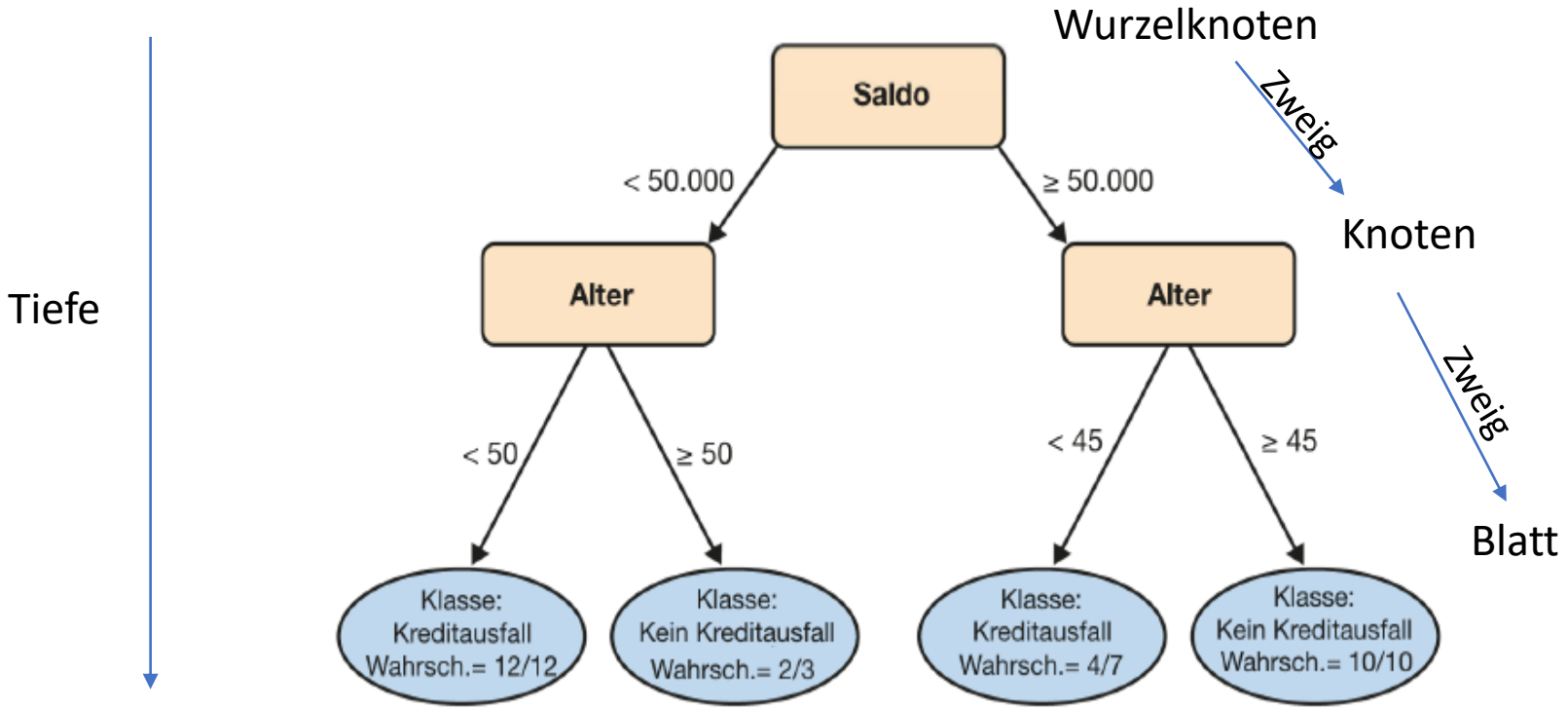
- Zweite grundlegende Idee: Eine Entscheidungsgrenze sollte den Abstand zu den nahesten Punkten maximieren. (Diese nahen Punkte sind die sogenannten „Support Vectors“ …)



https://de.wikipedia.org/wiki/Support_Vector_Machine

Klassifikation - Entscheidungsbaum

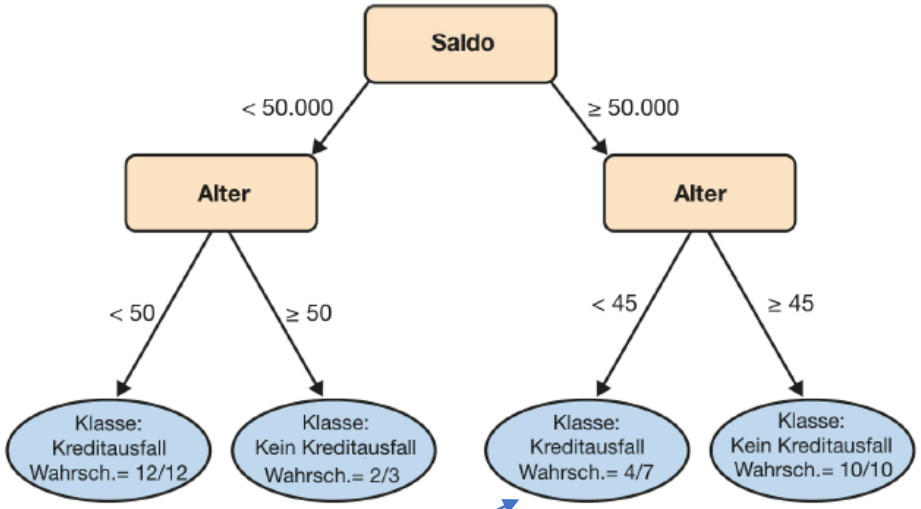
- Entscheidungsbaum zur Beurteilung der Kreditwürdigkeit von Kunden anhand von Merkmal Saldo und Alter



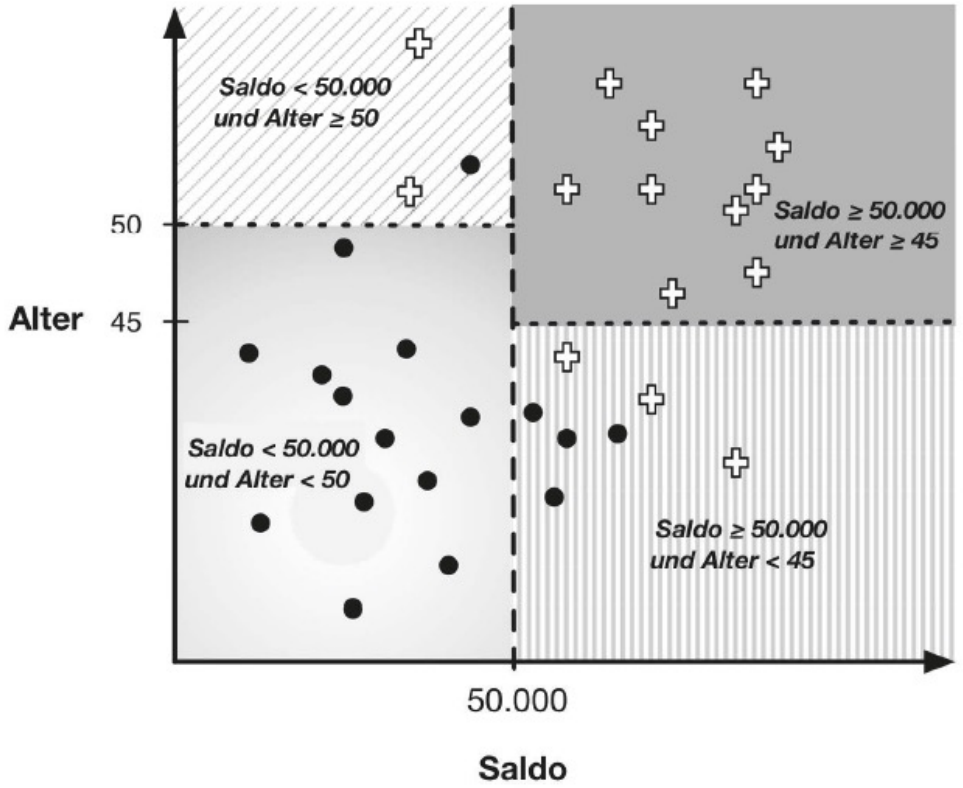
Foster P. , Fawcett, T., Data Science für Unternehmen, 2017, S.99

Klassifikation - Entscheidungsbaum

- Entscheidungsgrenzen sind keine Geraden oder Ebenen sondern Stufen/Bereiche



$$p(Kreditausfall) = \frac{samples\ Kreditausfall}{samples\ Gesamt\ in\ Bereich}$$

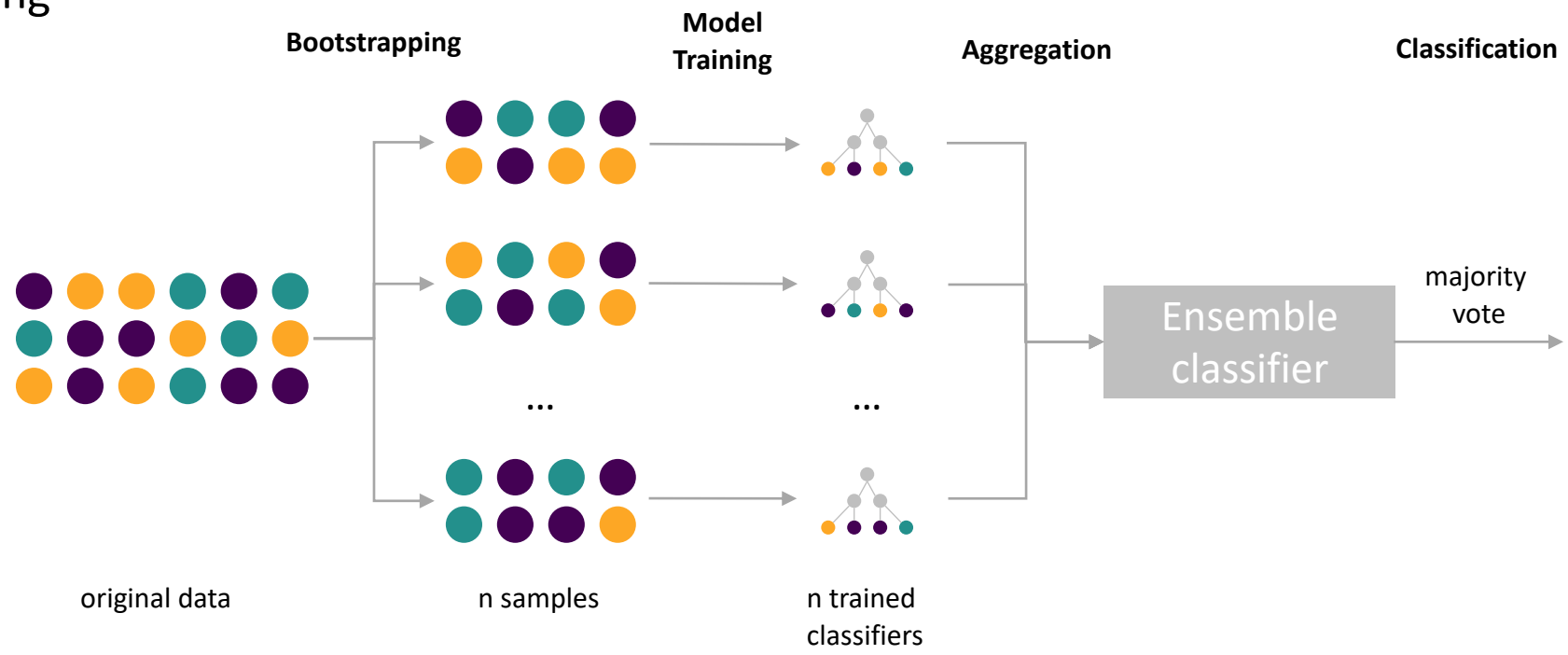


Foster P. , Fawcett, T., Data Science für Unternehmen, 2017, S.99

Ensemble Methoden - Bagging

- Um der Gefahr des Overfittings vorzubeugen können sogenannte Ensemble Methoden verwendet werden

Beispiel: Bagging



Küppers, P. , MW330 – L2B – Bias Variance Tradeoff, S.20; eigene Übersetzung