



Random Forest

Grundlagen des Maschinellen Lernens

Nils Schwenzfeier

19.05.2021
Prof. Dr. Volker Gruhn

1. Einführung
2. Decision Forest
 - 2.1. Training
 - 2.2. Inference
3. Boosting
 - 3.1. AdaBoost

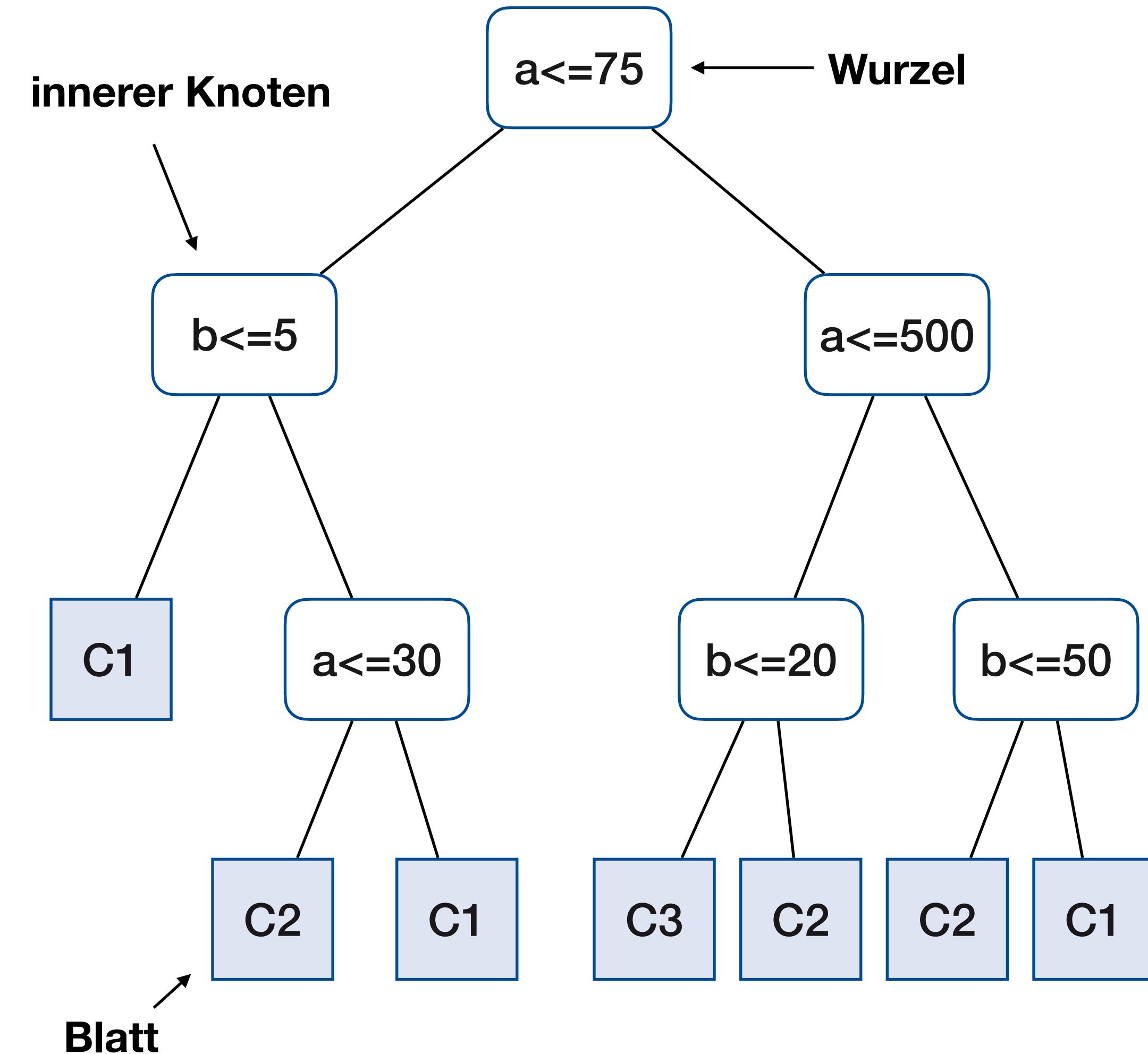
Einführung

1. Innerer Knoten (Split node)

- enthält Entscheidungsfunktion f_θ , welche die Beobachtungen in zwei Teilgruppen aufteilt

2. Blatt (Leaf/terminal node)

- modelliert die Ausgabe $P(Y|X)$ für einen Teil des Feature-Raums



- Forest = Ensemble von Bäumen
- Bäume sollten sich leicht von einander unterscheiden
- Bäume sollten unkorreliert sein

Decision Forest

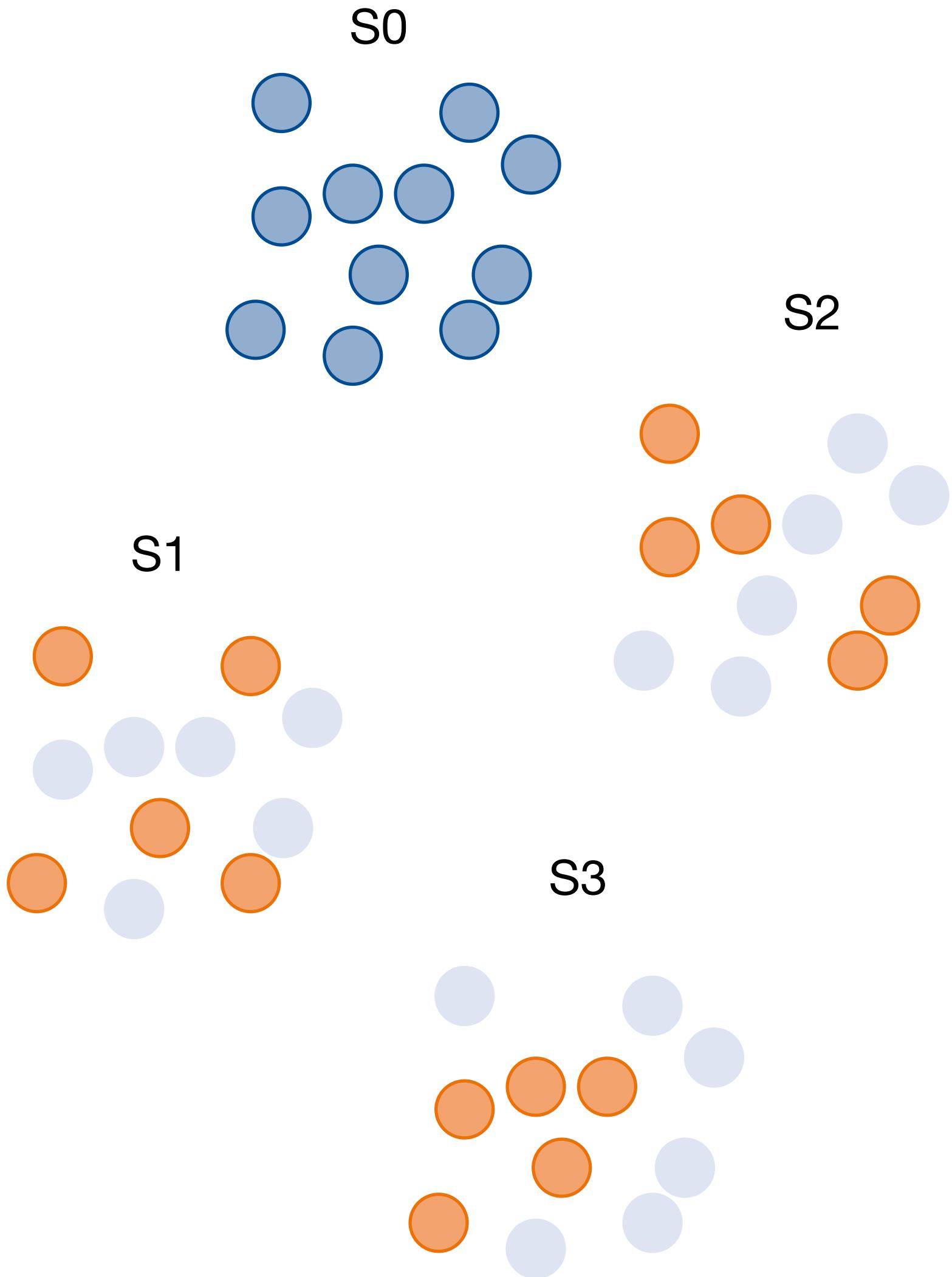
Formale Grundlage

- Wie erstellen wir unkorrelierte Bäume?
- Wie kombinieren wir deren Ausgaben?
- Wie viele Bäume benötigen wir?

Bagging

- ▶ Decision Forest
- ▶ Unkorrelierte Bäume

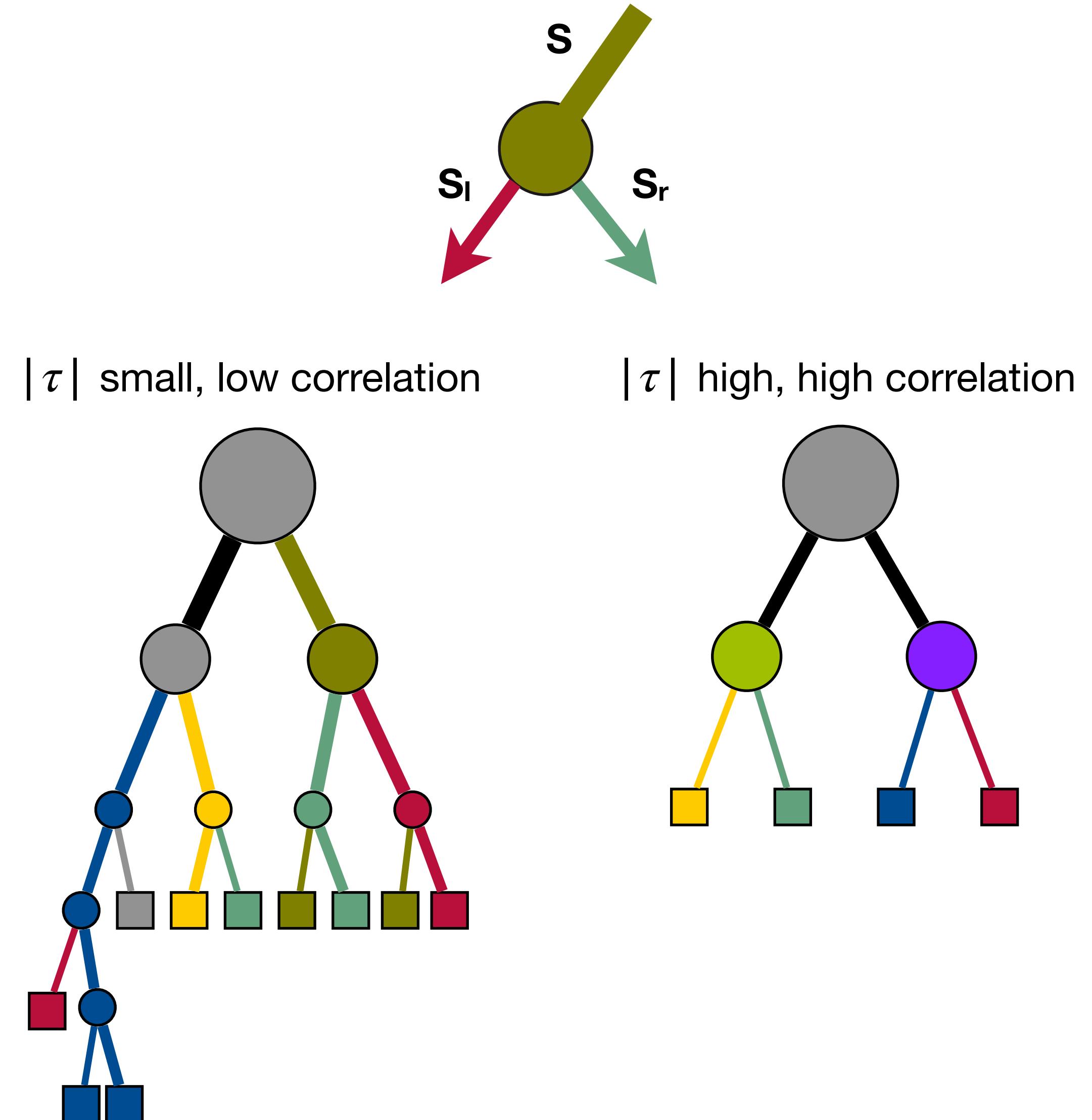
- Bagging = Bootstrap Aggregating
- Erstelle zufällige Teilmengen des Training-Sets
 S_0
- Trainiere jeden Baum auf **einem** dieser Teilmengen



Zufällige Knoten-Optimierung

- ▶ Decision Forest
- ▶ Unkorrelierte Bäume

- Set möglicher Parameter der Entscheidungsfunktion Γ
- Erstelle eine zufällige Teilmenge $\tau \subset \Gamma$
- Wähle die beste Funktion aus τ

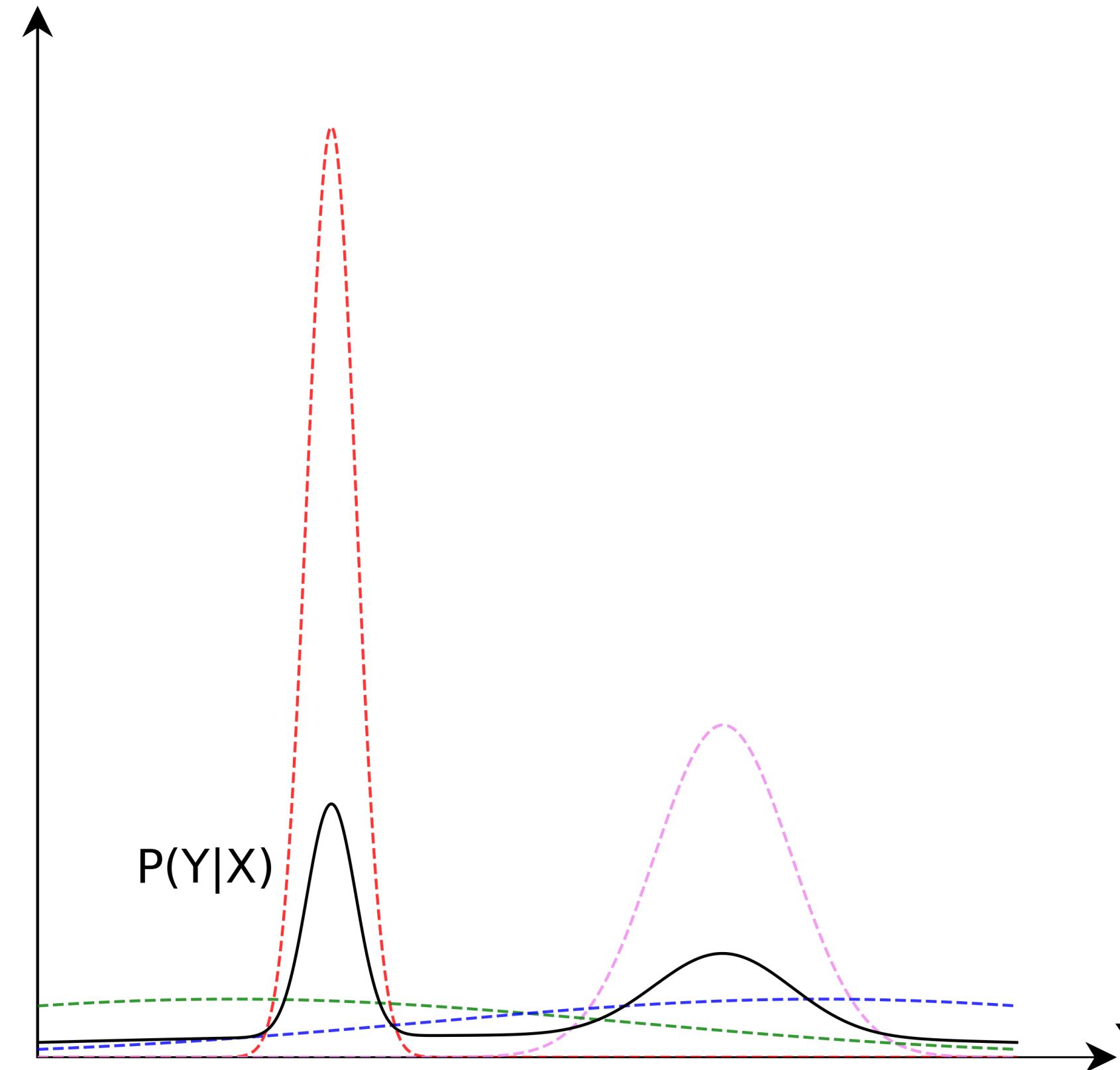


Ausgaben kombinieren

► Decision Forest

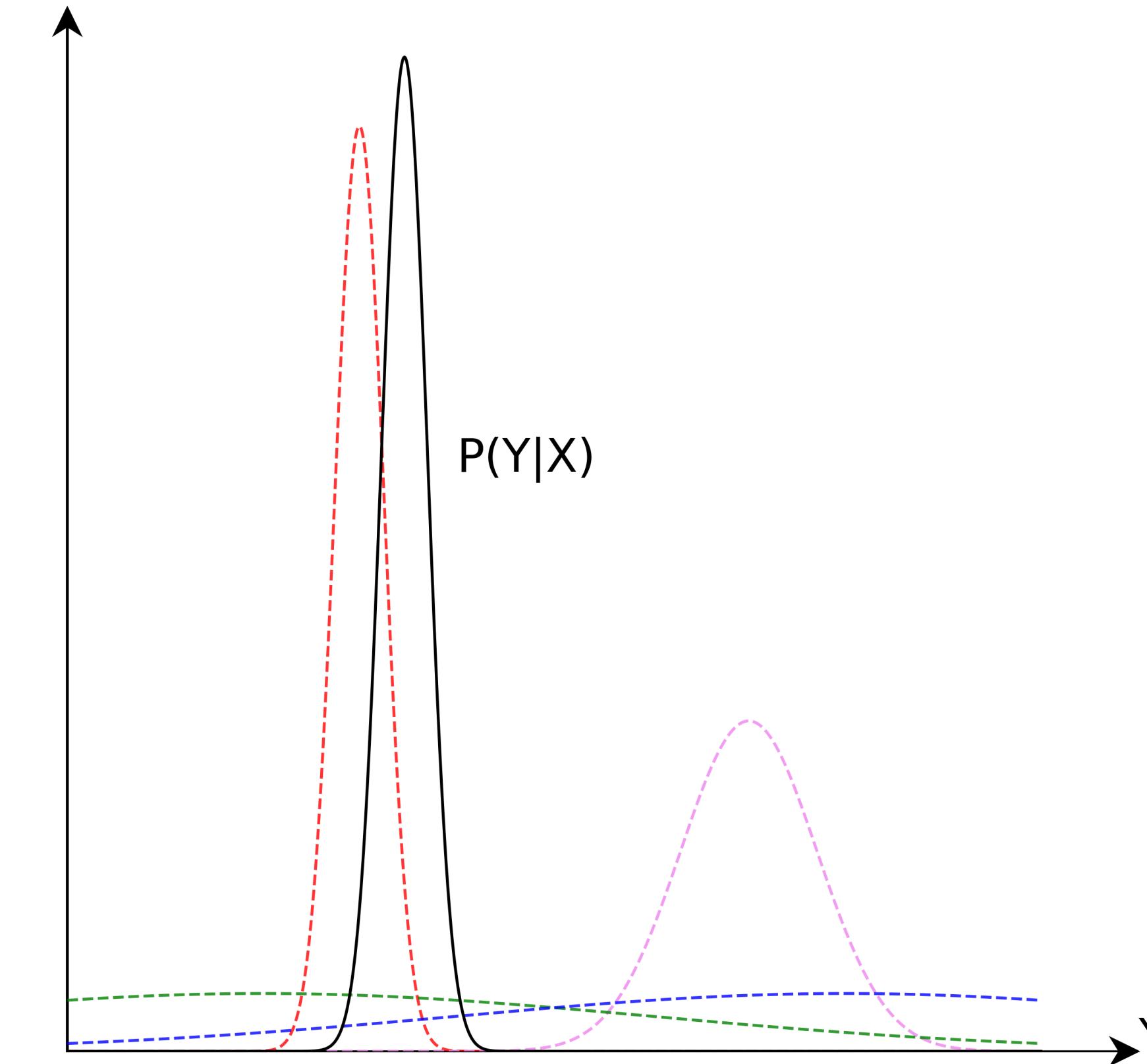
Summe der A-posteriori-Wahrscheinlichkeiten

$$P(Y|X) = \frac{1}{T} \sum_{t=1}^T P_t(Y|X).$$



Produkt der A-posteriori-Wahrscheinlichkeiten

$$P(Y|X) = \frac{1}{Z} \prod_{t=1}^T P_t(Y|X).$$



Komponenten

forest specific

Parameter

- Knoten-Entscheidungsfunktion f_θ
 - Parameter der Entscheidungsfunktion θ
 - Zielfunktion $I(S, S_l, S_r)$
 - Vorhersage-Modell der Blätter $P(Y|X)$
 - Forest Vorhersage-Modell
$$P(Y|X) = \frac{1}{T} \sum_{t=1}^T P_t(Y|X)$$
- Maximale Baumtiefe D
 - Minimale Population $MinPop$
 - Minimale Energie I_{min}
 - Bagging-Verhältnis $R = |S_0^t| / |S_0|$
 - Anzahl Versuche pro Knoten $|T|$
 - Anzahl der Bäume T

Classification und *Regression Forest* unterscheiden sich ausschließlich durch den Aufbau der verwendeten Bäume (s. Vorlesung Nicht-lineare Verfahren).

Random Forest in sklearn

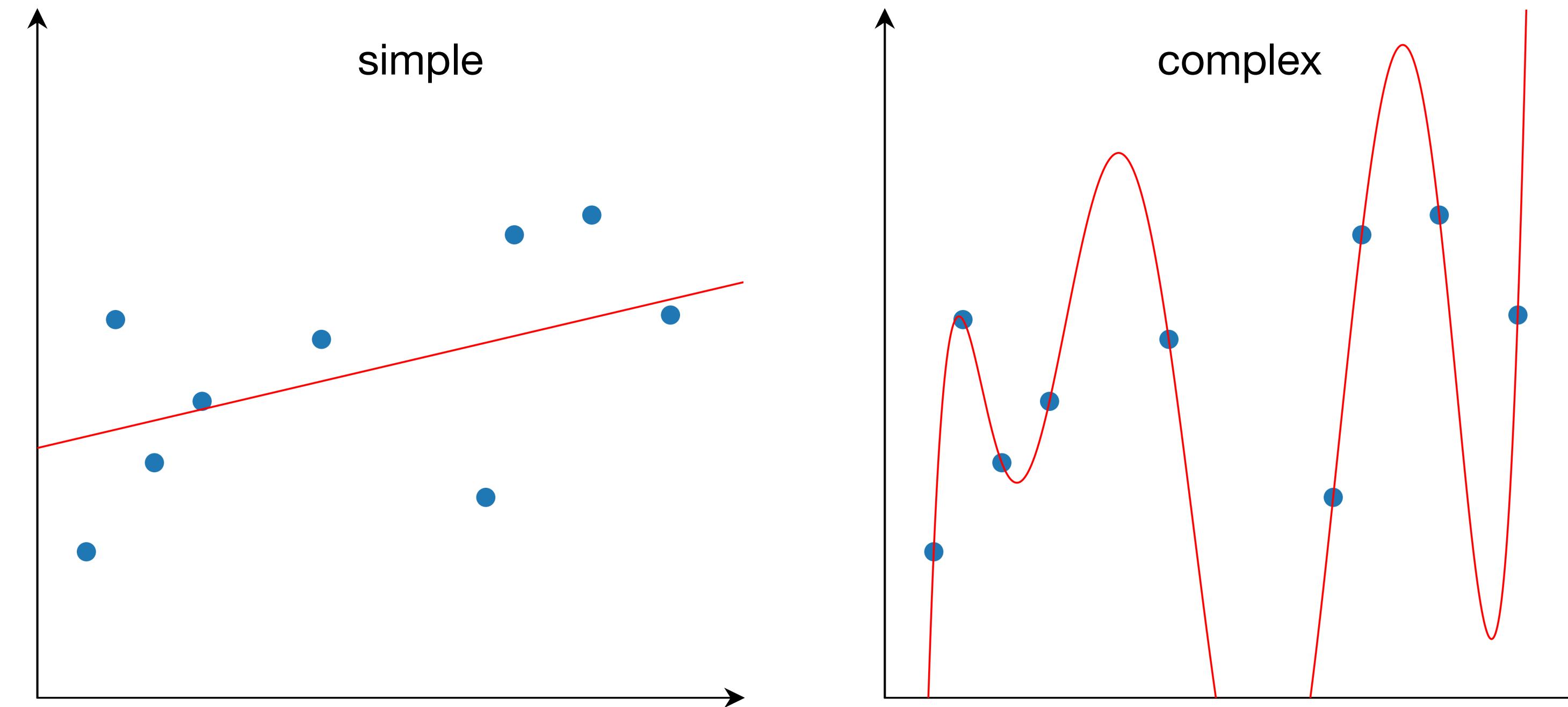
► Decision Forest



- Gleiches Interface wie andere Klassifikations-/Regressions-Modelle
- Neben den Hyperparametern der Entscheidungsbäume müssen zusätzlich die Einstellungen für das Ensemble konfiguriert werden

```
from sklearn.ensemble import RandomForestClassifier  
  
clf = RandomForestClassifier(n_estimators=100,  
                             bootstrap=True,  
                             criterion='gini',  
                             max_depth=2)  
  
clf.fit(X, y)  
y_pred = clf.predict(X)
```

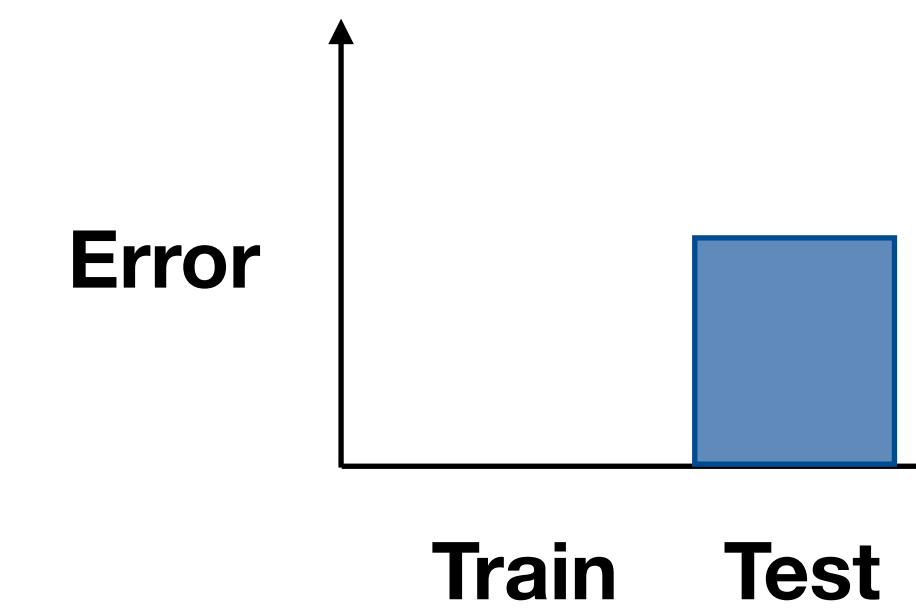
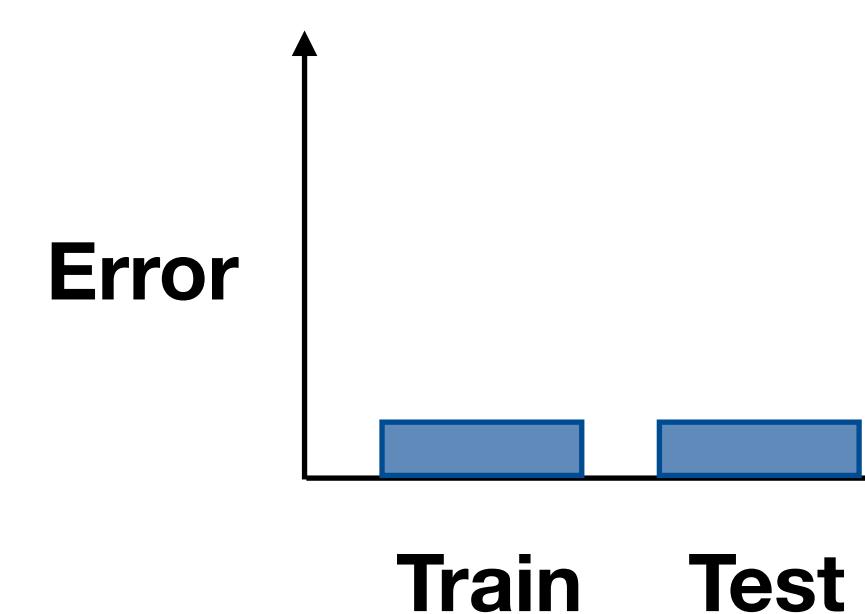
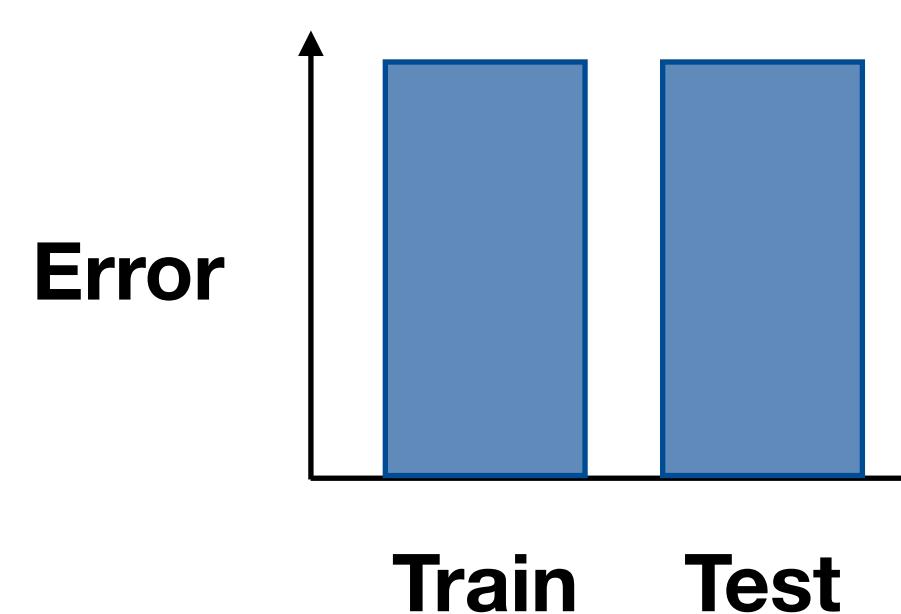
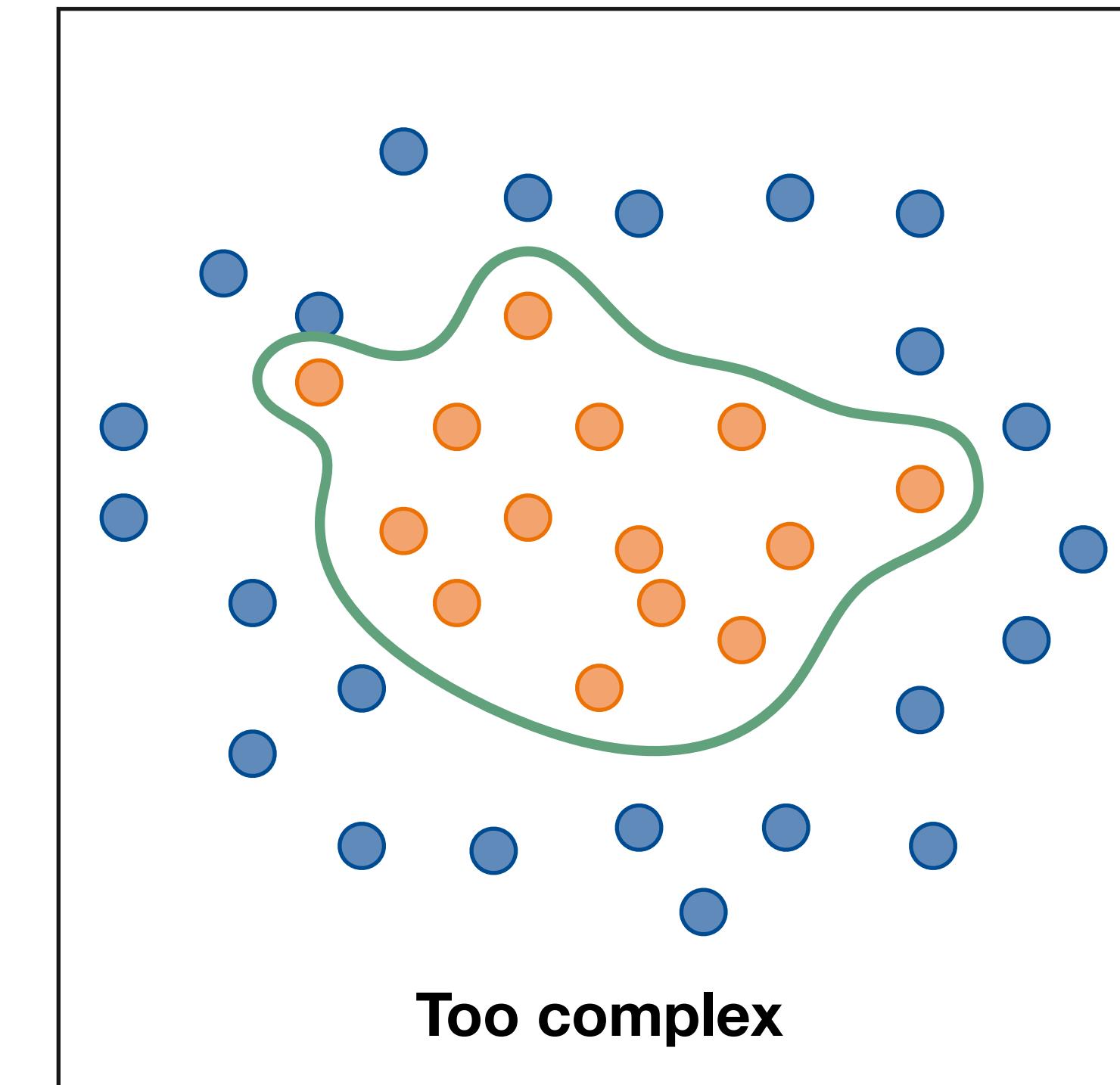
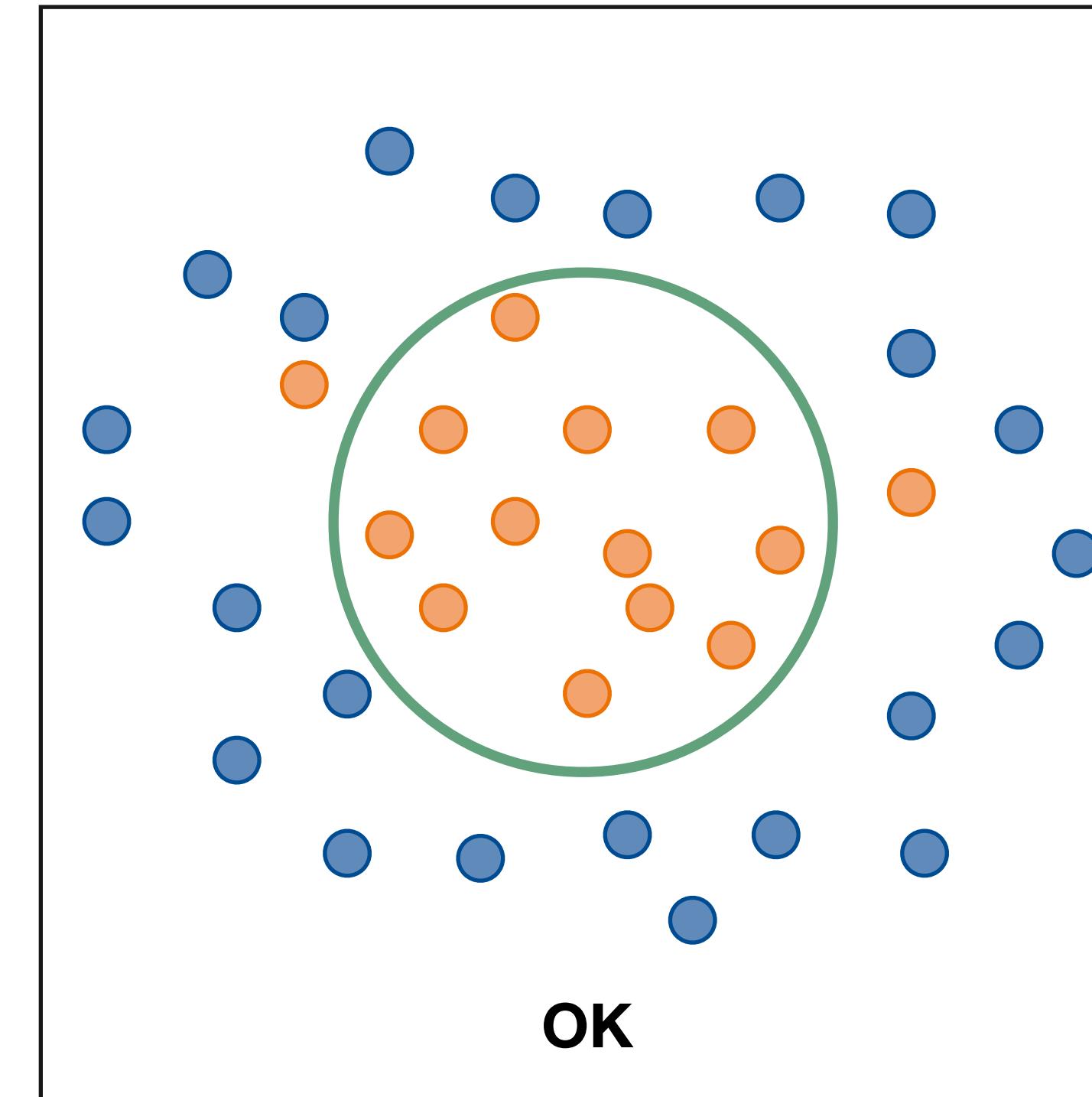
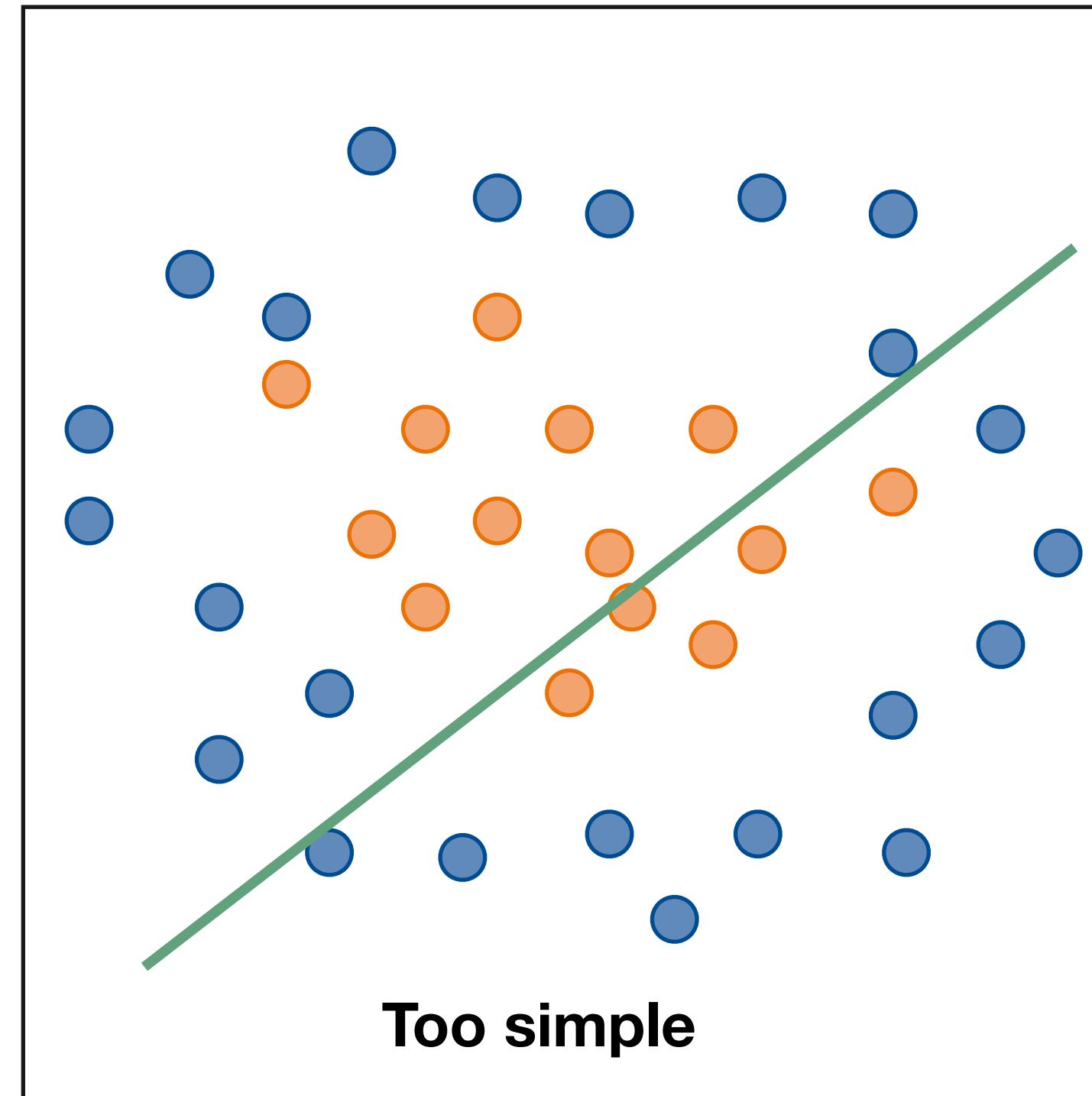
Boosting



- Einfaches Modell: hoher Bias, geringe Varianz = Underfitting
- Komplexes Modell: geringer Bias, hohe Varianz = Overfitting
- Ziel: geringer Bias & geringe Varianz oder ein “guter” Kompromiss

Das “richtige” Modell wählen

▶ Boosting

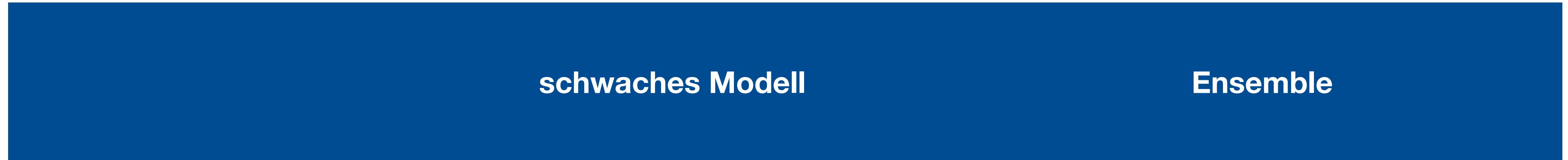


- Idee: konstruiere ein starkes Klassifikations-Modell aus schwachen Klassifikations-Modellen
- Wie verknüpfen wir die Ergebnisse der verschiedenen Modelle?
- Reduzieren sich dadurch Bias & Varianz?

Varianz reduzieren: Majority Voting

► Boosting

Für ein Set von M schwachen Modellen (f_1, \dots, f_M) , baue ein Ensemble-Modell: $\hat{f}(x) = \frac{1}{M} \sum_{m=1}^M f_m(x)$



Vorhersage-Fehler

$$\epsilon_m(x) = y(x) - f_m(x)$$

$$\hat{\epsilon}(x) = y(x) - \frac{1}{M} \sum_{m=1}^M f_m(x)$$

Durchschnitts-Summe der quadrierten Fehler

$$E_m = \mathbb{E}_x [\epsilon_m(X)^2]$$

$$\hat{E} = \mathbb{E}_x \left[\left(\frac{1}{M} \sum_{m=1}^M \epsilon_m(x) \right)^2 \right]$$

Durchschnittlicher Fehler

$$E_{AV} = \frac{1}{M} \sum_{m=1}^M \mathbb{E}_x [\epsilon_m(X)^2]$$

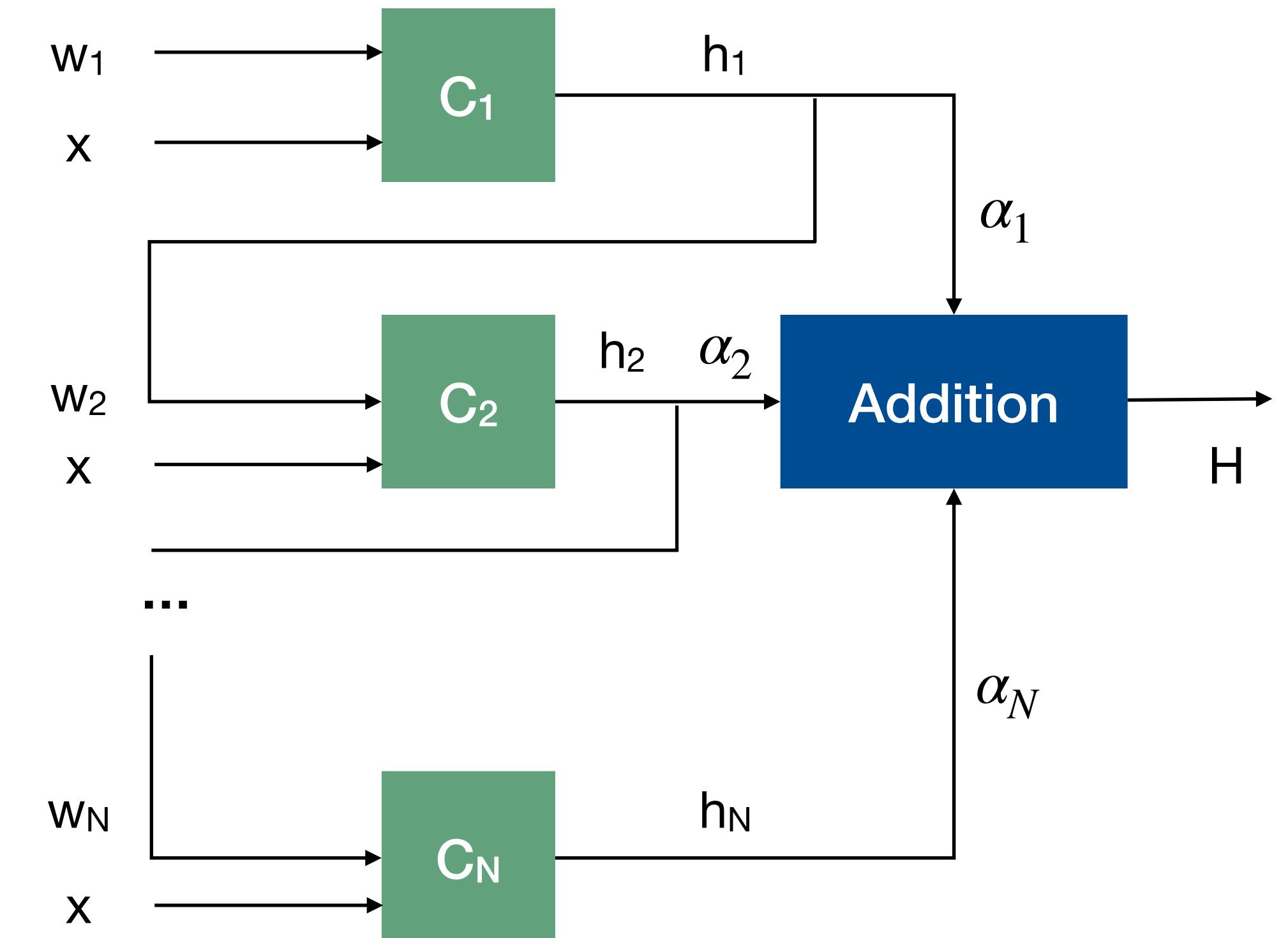
falls $\mathbb{E}_x [\epsilon_m(x)] = \mathbb{E}_x [\epsilon_m(x)\epsilon_j(x)] = 0$

Unbiased $\hat{E} = \frac{1}{M} E_{AV}$ **Covariance**

AdaBoost (Freund & Schapire 1995)

▶ Boosting

- Adaptive Boosting
- Additives Modell
- Modifiziere das Datenset iterativ, um auf fehlklassifizierte Datensätze zu fokussieren
- Gewichte die Vorhersagen basierend auf der Performanz der Klassifikatoren

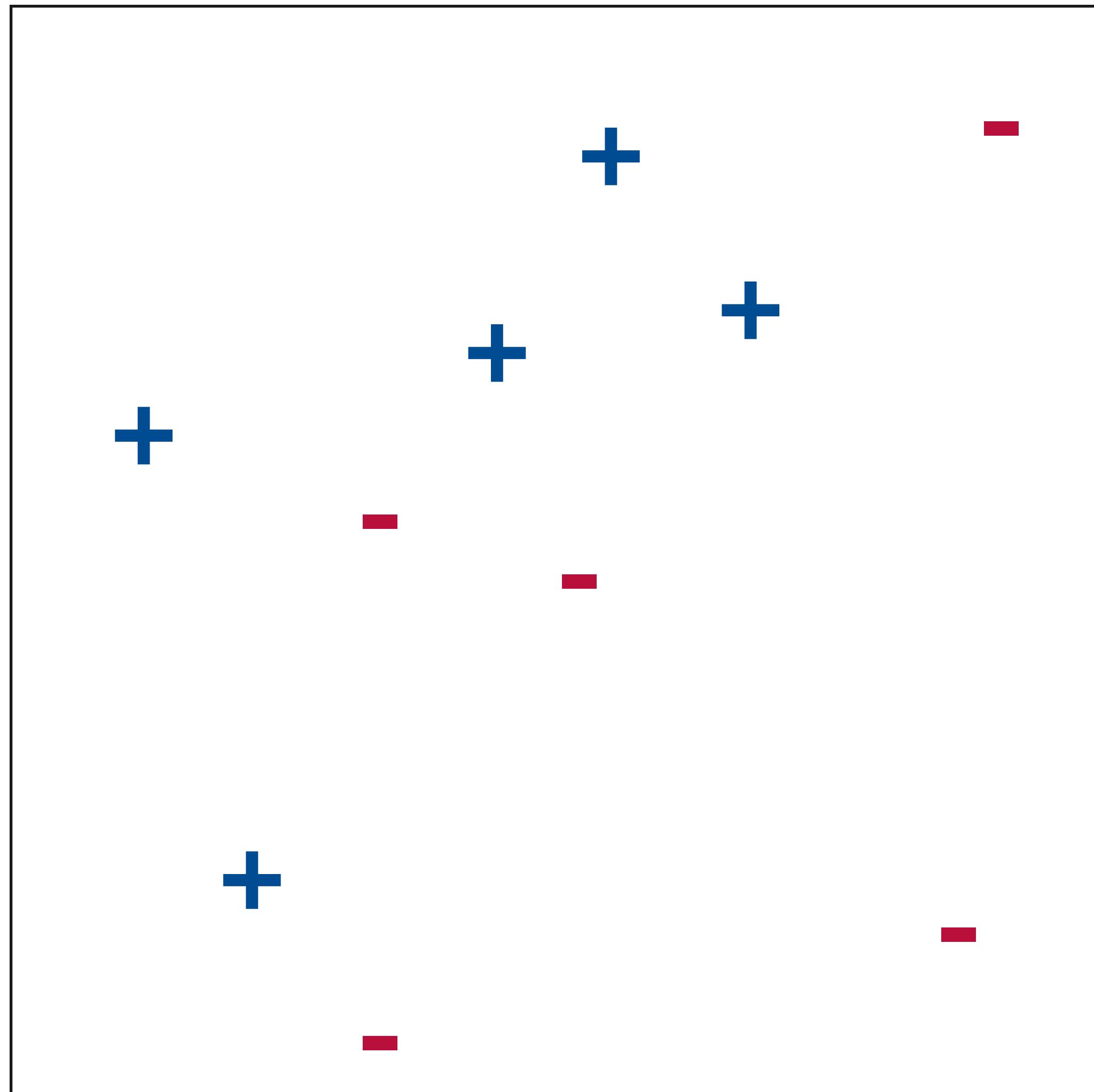


Beispiel

- ▶ Boosting
- ▶ AdaBoost

Schwache Modelle: decision stumps

Jeder Datenpunkt mit Label $y_i \in \{+1, -1\}$ und
Gewicht $w_i = 1$



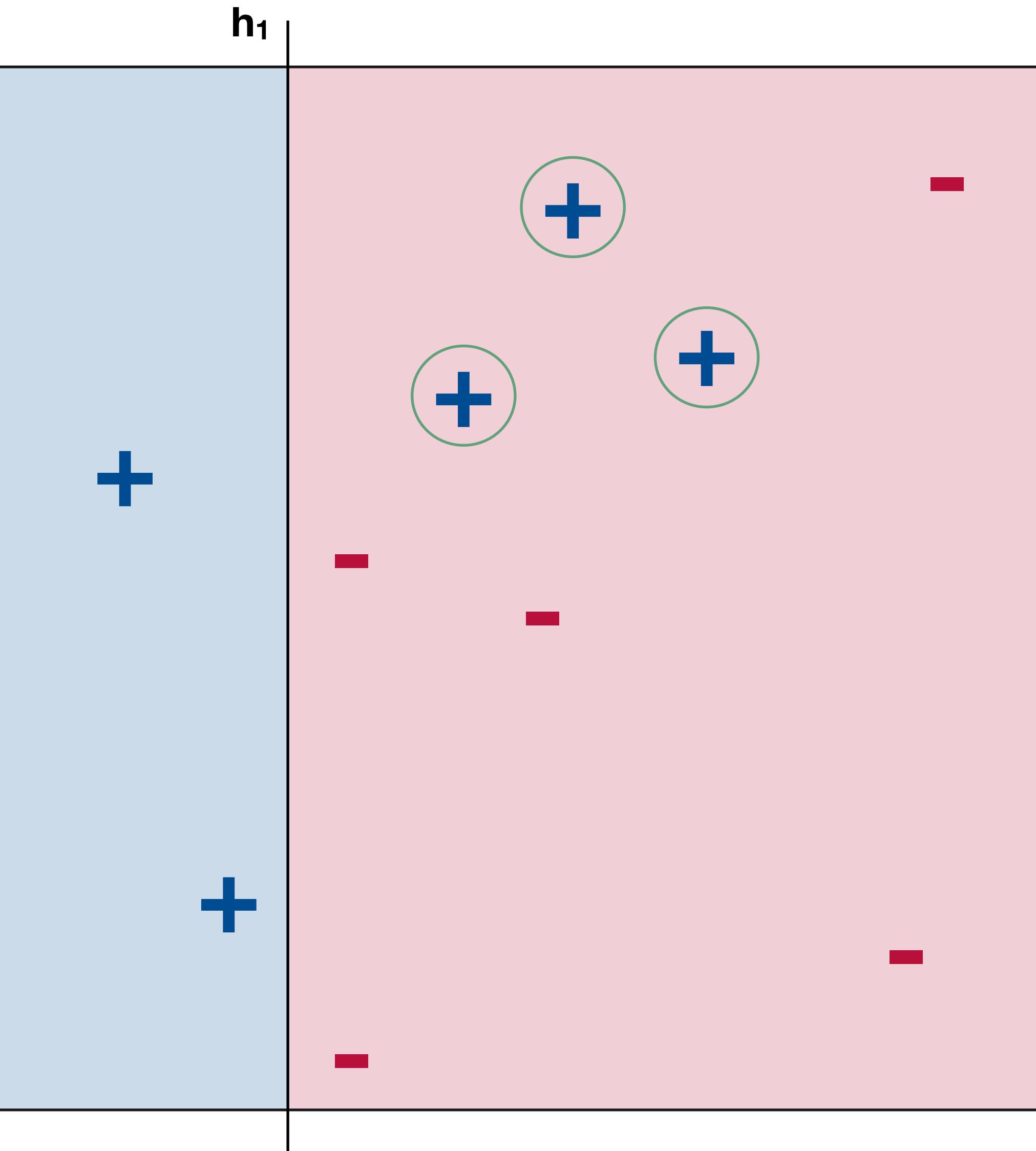
AdaBoost - Beispiel

- ▶ Boosting
- ▶ AdaBoost

Schwache Modelle: decision stumps

Jeder Datenpunkt mit Label $y_i \in \{+1, -1\}$ und
Gewicht $w_i = 1$

- **Finde die beste Trennlinie**



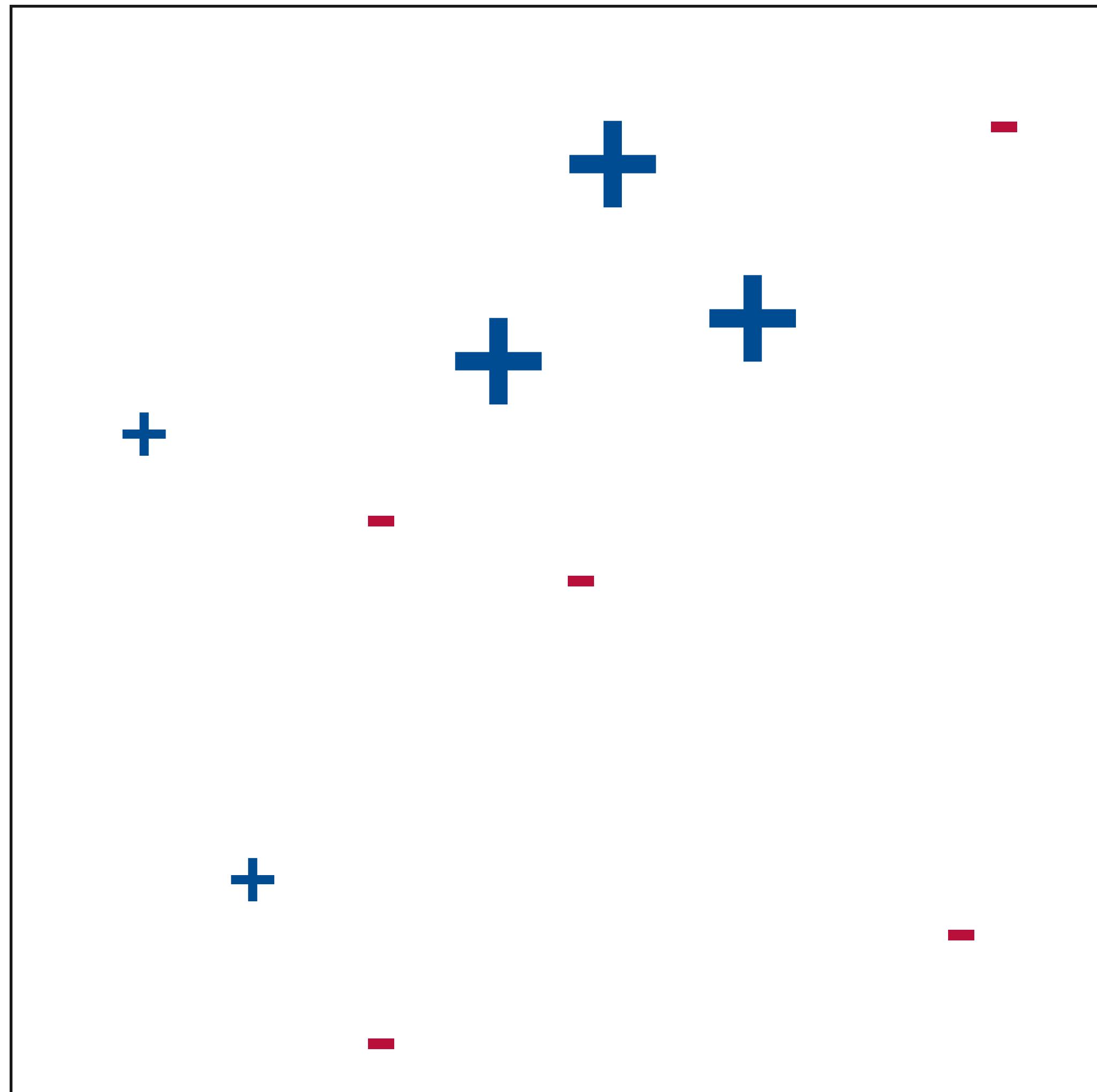
AdaBoost - Beispiel

- ▶ Boosting
- ▶ AdaBoost

Schwache Modelle: decision stumps

Jeder Datenpunkt mit Label $y_i \in \{+1, -1\}$ und
Gewicht $w_i = 1$

- Finde die beste Trennlinie
- **Aktualisiere die Gewichte:**
 $w_i \leftarrow w_i \exp(-y_i H_i)$



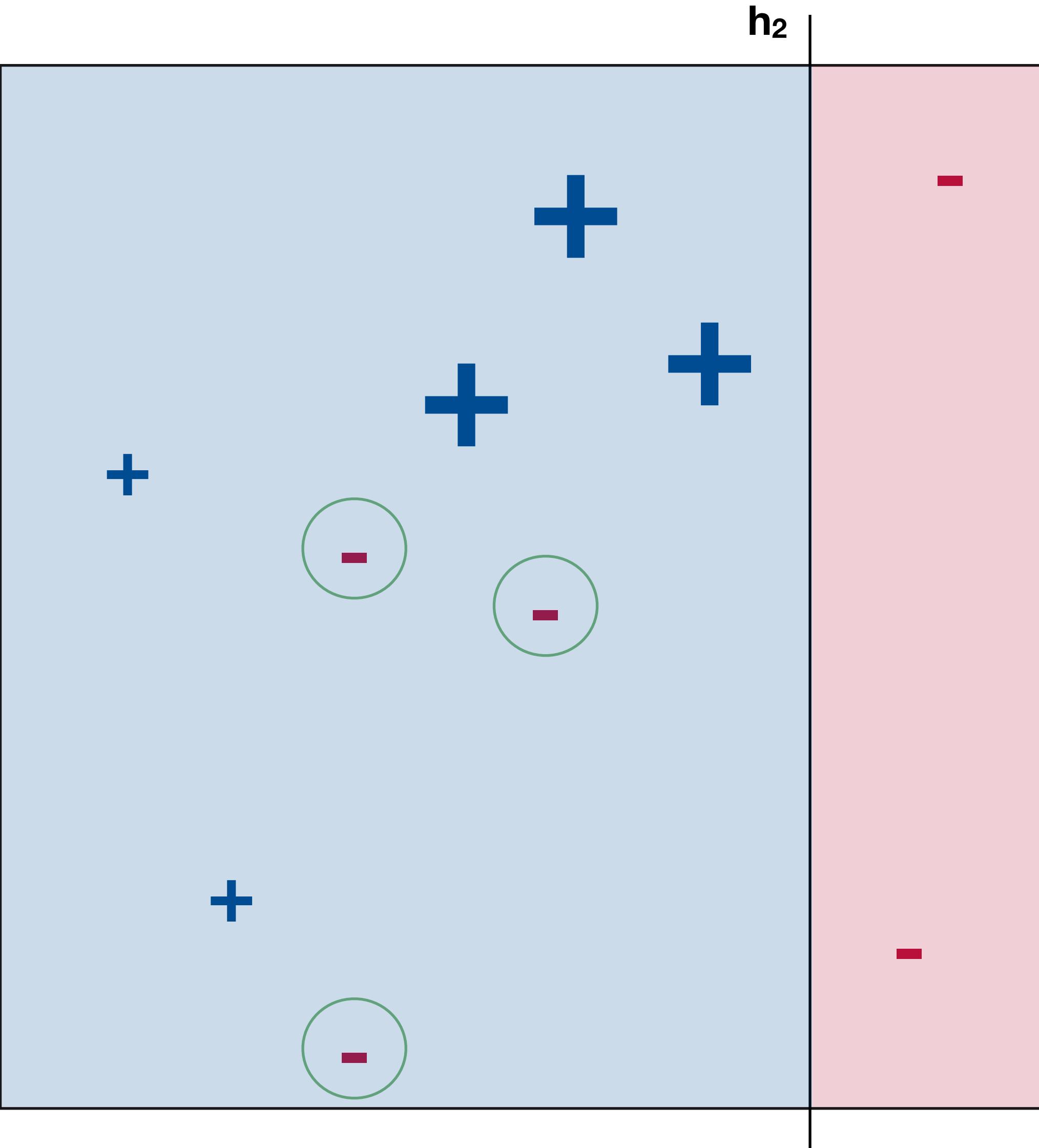
AdaBoost - Beispiel

- ▶ Boosting
- ▶ AdaBoost

Schwache Modelle: decision stumps

Jeder Datenpunkt mit Label $y_i \in \{+1, -1\}$ und
Gewicht $w_i = 1$

- **Finde die beste Trennlinie**
- Aktualisiere die Gewichte:
 $w_i \leftarrow w_i \exp(-y_i H_i)$



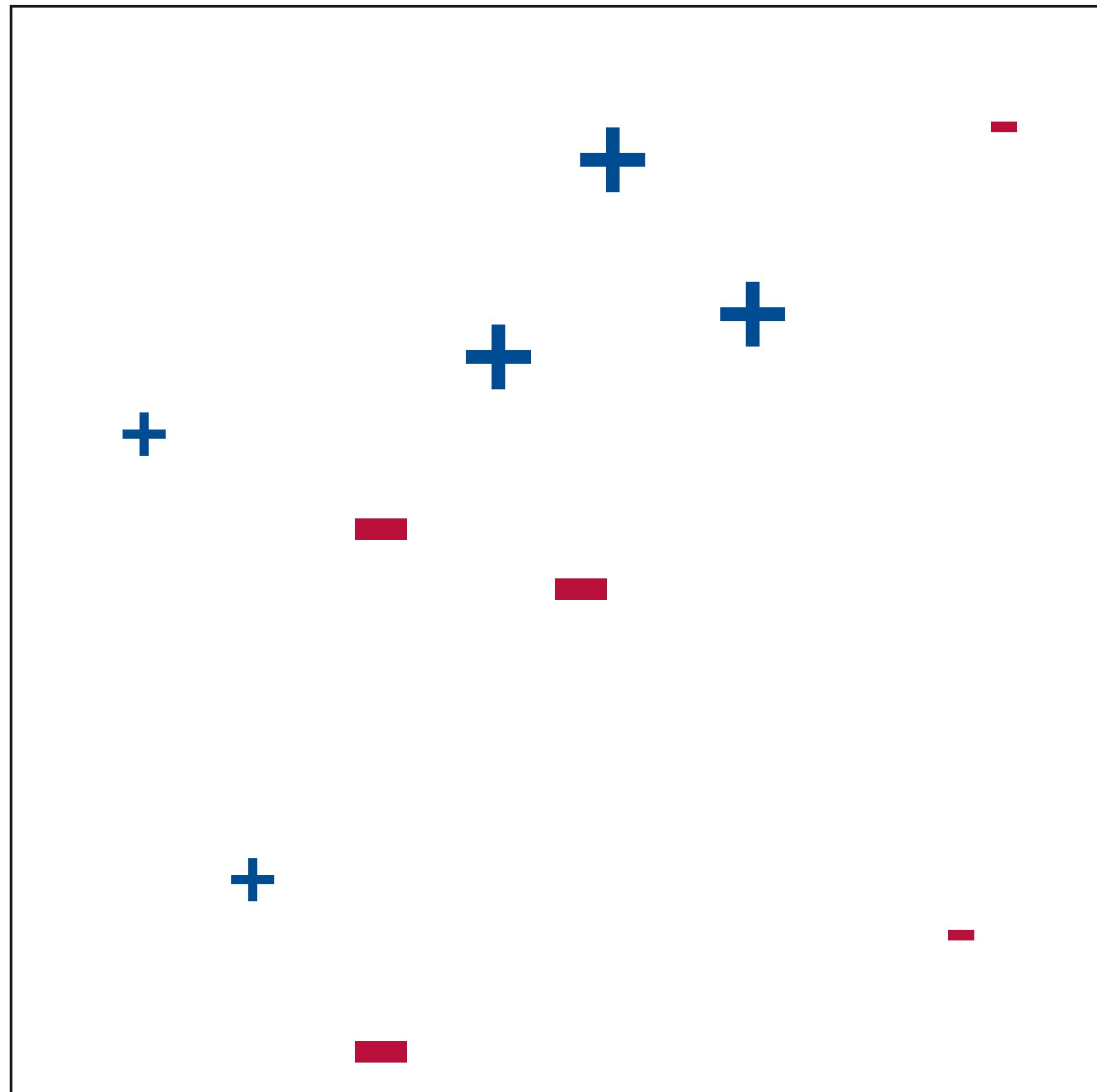
AdaBoost - Beispiel

- ▶ Boosting
- ▶ AdaBoost

Schwache Modelle: decision stumps

Jeder Datenpunkt mit Label $y_i \in \{+1, -1\}$ und
Gewicht $w_i = 1$

- Finde die beste Trennlinie
- **Aktualisiere die Gewichte:**
 $w_i \leftarrow w_i \exp(-y_i H_i)$



AdaBoost - Beispiel

- ▶ Boosting
- ▶ AdaBoost

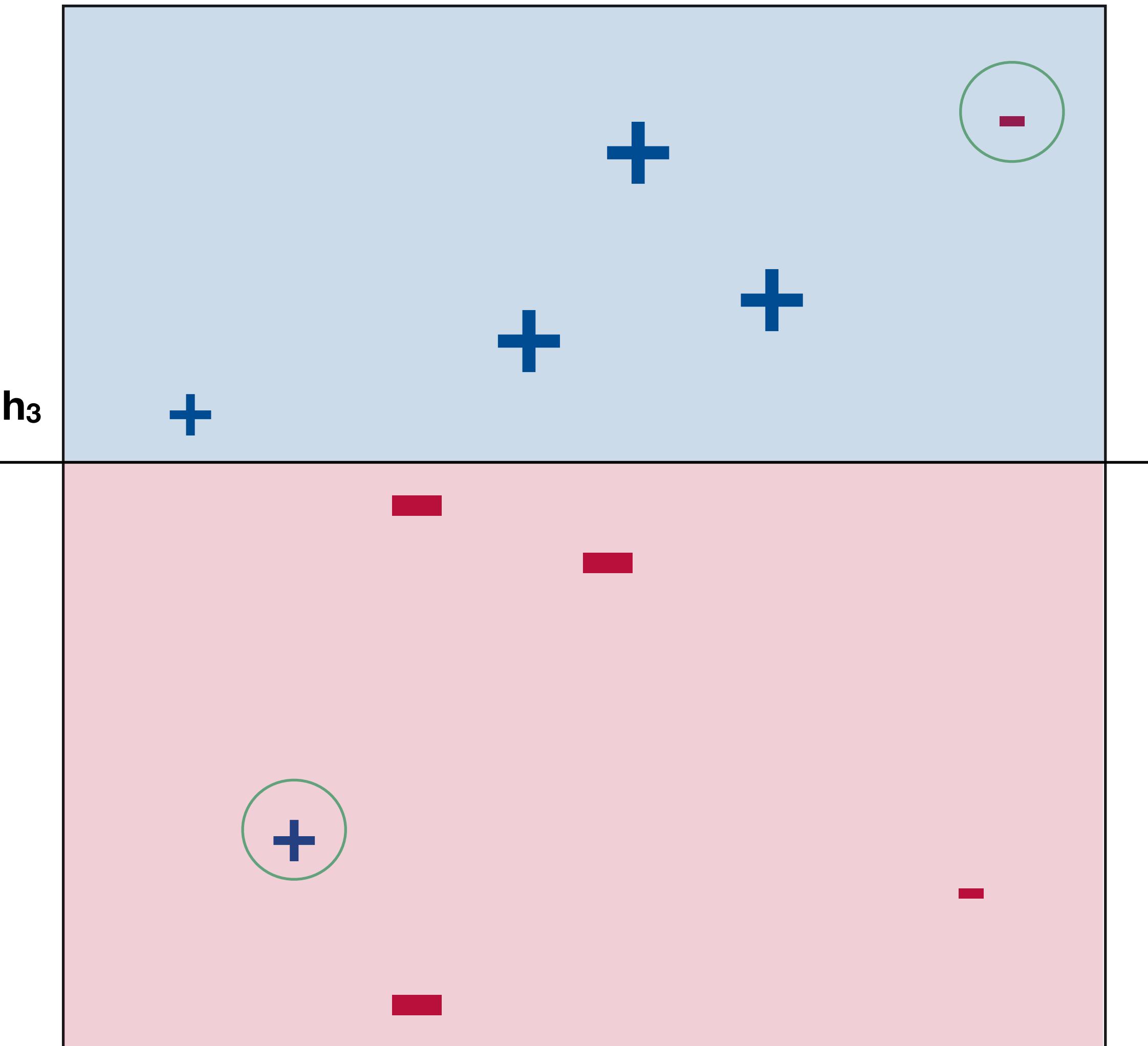
Schwache Modelle: decision stumps

Jeder Datenpunkt mit Label $y_i \in \{+1, -1\}$ und
Gewicht $w_i = 1$

- **Finde die beste Trennlinie**

- Aktualisiere die Gewichte:

$$w_i \leftarrow w_i \exp(-y_i H_i)$$



AdaBoost - Beispiel

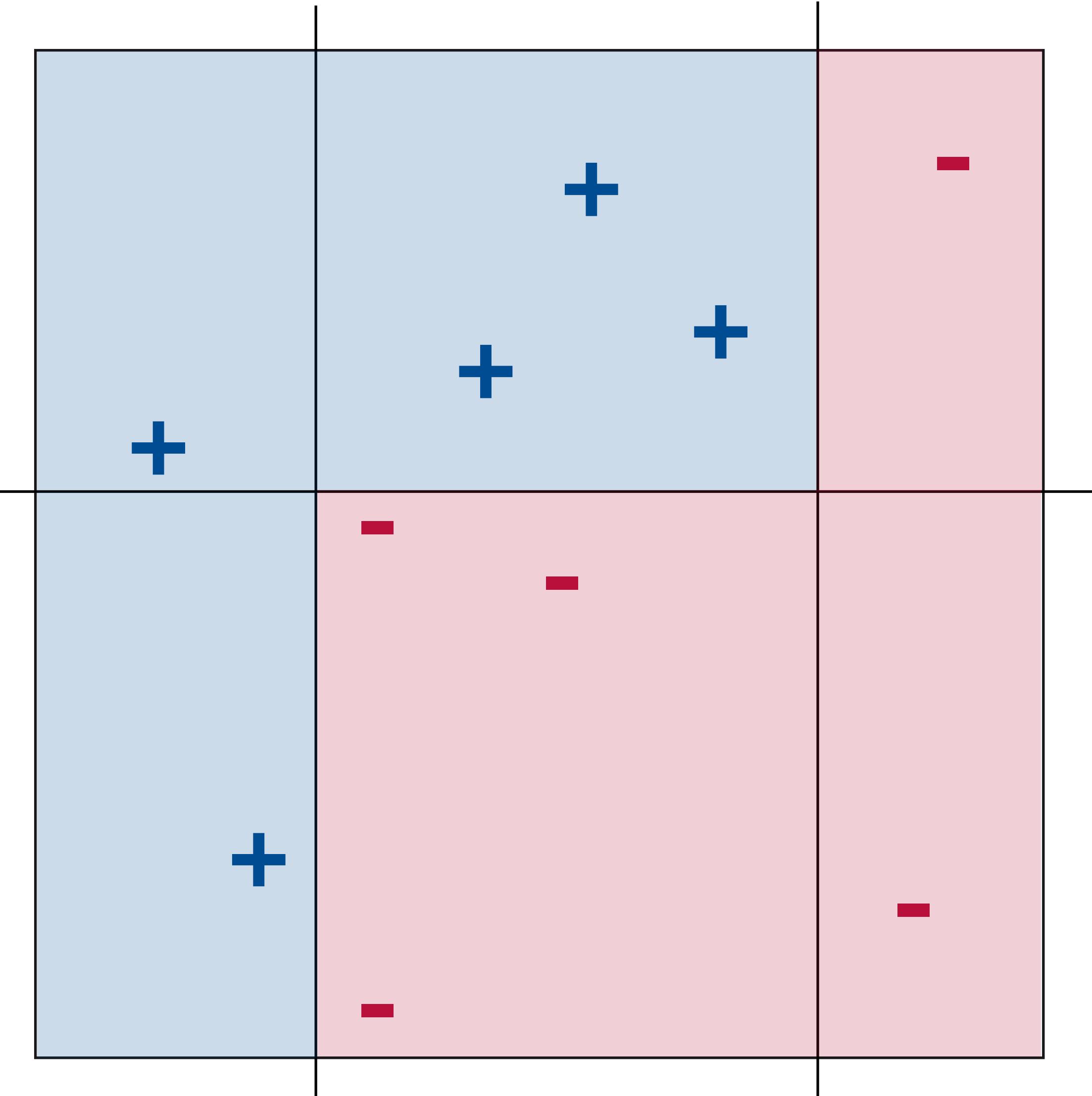
- ▶ Boosting
- ▶ AdaBoost

Schwache Modelle: decision stumps

Jeder Datenpunkt mit Label $y_i \in \{+1, -1\}$ und
Gewicht $w_i = 1$

- Finde die beste Trennlinie
- Aktualisiere die Gewichte:
 $w_i \leftarrow w_i \exp(-y_i H_i)$

$$H_{final} = \text{sign} \left(\alpha_1 \begin{array}{|c|c|} \hline \text{light blue} & \text{pink} \\ \hline \end{array} + \alpha_2 \begin{array}{|c|c|} \hline \text{light blue} & \text{pink} \\ \hline \end{array} + \alpha_3 \begin{array}{|c|c|} \hline \text{light blue} & \text{pink} \\ \hline \end{array} \right)$$



AdaBoost - Algorithmus

- ▶ Boosting
- ▶ AdaBoost

Datenset: $(x_i, y_i)_{i=1}^N$, $x_i \in X$, $y_i \in \{-1, 1\}$

Familie der Klassifikationsmodelle: $\mathcal{H} \subset \{-1, 1\}^X$

Für $t = 1 \dots T$

- Finde $h_t = \arg \min_{h \in \mathcal{H}} \epsilon(h_t)$, wobei $\epsilon(h_t) = \frac{\sum_{i=1}^N w_i^t [y_i \neq h_t(x_i)]}{\sum_{i=1}^N w_i^t}$

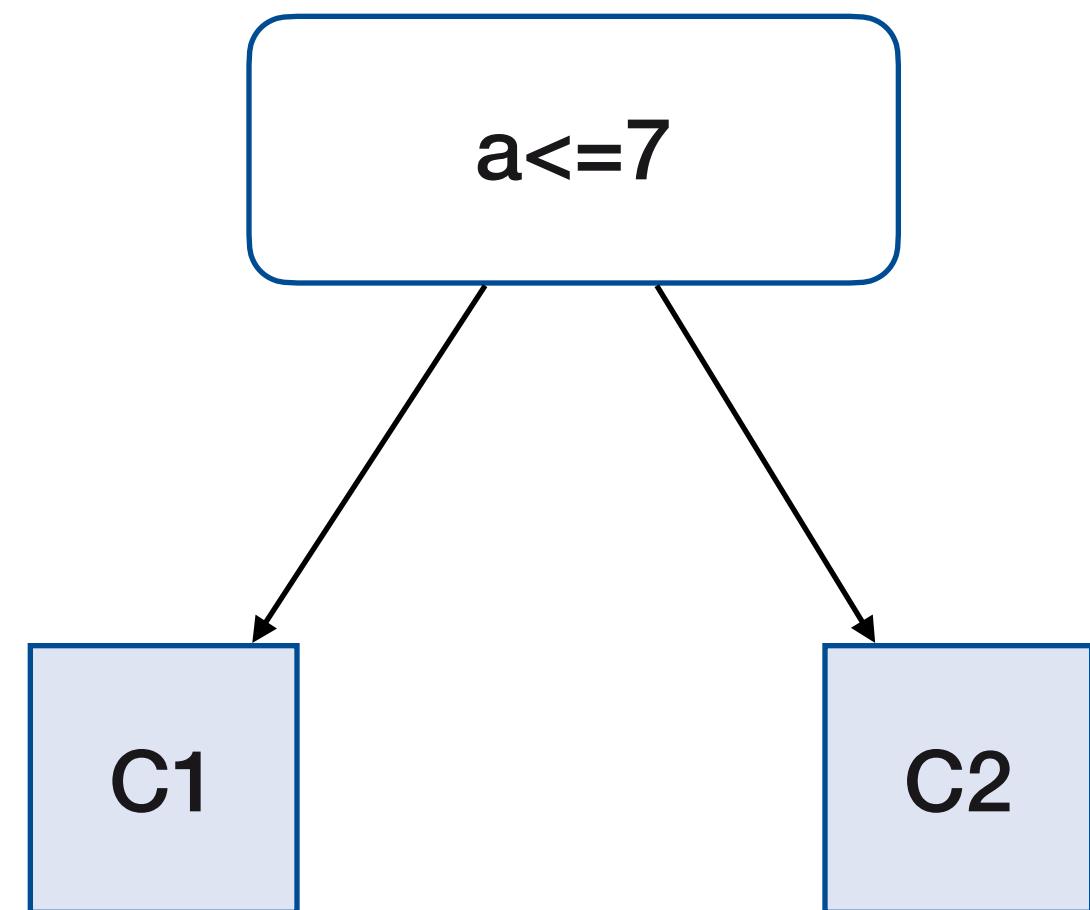
- Aktualisiere die Verteilung: $w_i^{t+1} = \frac{w_i^t}{Z_t} \exp(-\alpha_t y_i h_t(x_i))$

$$Z_t = \sum_{i=1}^N w_i^t \exp(-\alpha_t y_i h_t(x_i))$$

$$\alpha_t = \frac{1}{2} \log \frac{1 - \epsilon_t}{\epsilon_t}$$

Finales Modell: $H(x) = \text{sign} \left(\sum_{t=1}^N \alpha_t h_t(x) \right)$

- Verbessert die Klassifikations-Genauigkeit
- Kann mit vielen verschiedenen Klassifizierungs-Modellen genutzt werden (Meta-Algorithmus)
- Nicht anfällig gegenüber Overfitting
- AdaBoost reduziert den Bias und die Varianz!



- Entscheidungsbaum mit nur einem Entscheidungsknoten
- Hoher Bias, geringe Varianz

AdaBoost in sklearn

▶ Boosting



- AdaBoost ist ein Meta-Algorithmus
 - Ein Klassifikationsmodell wird als Eingabe übergeben
 - AdaBoostClassifier in sklearn verwendet Decision Stumps (DecisionTreeClassifier mit max_depth=1)

```
from sklearn.ensemble import AdaBoostClassifier
from sklearn.tree import DecisionTreeClassifier

child_clf = DecisionTreeClassifier(max_depth=1)

clf = AdaBoostClassifier(n_estimators=100,
                        base_estimator=child_clf)

clf.fit(X, y)
clf.predict(X)
```