

# Ensemble Methoden

Ensemble-Methoden versuchen, mehrere „schwache“ Lerner zu einem hoffentlich besseren Lerner zusammenzusetzen.

Wir betrachten die beiden wichtigsten Ensemble-Methoden,

## **Bagging** und **Boosting**

Wir suchen nach einer gemeinsamen Erklärung für den häufig beobachteten Erfolg dieser beiden Methoden.

# Der lineare Abschluss einer Hypothesenklasse

# Linearkombinationen von Hypothesen

Sei  $\mathcal{H}$  eine Hypothesenklasse von Funktionen  $h : X \rightarrow \{-1, 1\}$ .

Wir definieren den **linearen Abschluss** von  $\mathcal{H}$  durch

$$\hat{\mathcal{H}} := \left\{ \sum_{i=1}^p \alpha_i h_i : p \in \mathbb{N}, h_1, \dots, h_p \in \mathcal{H}, \alpha_1, \dots, \alpha_p \in \mathbb{R} \right\}$$

Die Hypothesenklasse des linearen Abschlusses ist

$$\text{sign}(\hat{\mathcal{H}}) := \{ \text{sign}(g) : g \in \hat{\mathcal{H}} \}.$$

Wie viele Freiheitsgrade besitzt die Hypothesenklasse des linearen Abschlusses?

# Die VC-Dimension von $\text{sign}(\hat{\mathcal{H}})$

Sei  $\mathfrak{I}$  die Hypothesenklasse, die für jedes Intervall  $I \subseteq [0, 1]$  die Funktion  $h_I : [0, 1] \rightarrow \{-1, 1\}$  mit  $h_I(x) = 1 \iff x \in I$  besitzt.

1.  $g_I := (h_I + h_{[0,1]})/2$  ist die Indikatorfunktion des Intervalls  $I$ .
2. Für eine endliche Vereinigung  $J = I_1 \dot{\cup} \dots \dot{\cup} I_k$  von Intervallen gilt

$$x \in J \iff \text{sign}\left(g_{I_1}(x) + \dots + g_{I_k}(x) - \frac{g_{[0,1]}(x)}{2}\right) = 1.$$

3.  $\implies \text{VC}(\mathfrak{I}) = 2$ , aber  $\text{VC}(\text{sign}(\hat{\mathfrak{I}})) = \infty$ .

Unter welchen Bedingungen kann man Hypothesen aus  $\hat{\mathcal{H}}$  (mit wenigen Beispielen) lernen?

# Zur Erinnerung: Margin- und Hinge-Loss

$D$  sei eine Verteilung über  $X$  und  $f$  eine unbekannte Zielfunktion.

$$\text{Loss}^{S,\rho}(h) := \frac{1}{s} \cdot \sum_{i=1}^s \Phi_{\rho} \left( f(x_i)h(x_i) \right) \text{ mit } \Phi_{\rho}(z) := \begin{cases} 0 & \rho < z, \\ 1 - \frac{z}{\rho} & 0 \leq z \leq \rho, \\ 1 & z < 0 \end{cases}$$

ist der **Margin-Loss**. Der **erwartete Hinge Loss** einer Hypothese  $h$  ist

$$\text{Loss}_D(h) = \mathbb{E}_{x \sim D} [\Phi_{\rho}(f(x)h(\phi(x)))].$$

Wenn der empirische Margin-Loss klein ist: Die für erfolgreiches Lernen notwendige Beispielzahl ist hoffentlich auch klein.

# Beispielzahl für den linearen Abschluss

$\mathcal{H}$  sei eine Klasse von Hypothesen mit Wertebereich  $\{-1, 1\}$ .

Es gelte  $VC(\mathcal{H}) = d$ . Dann gilt für alle  $\rho > 0$ ,  $\delta > 0$ ,  
alle Verteilungen  $D$  über  $X$  und alle Hypothesen  $\hat{h} \in \hat{\mathcal{H}}$

$$\underbrace{\text{Loss}_D(\hat{h})}_{\text{Erw. Hinge-Loss}} \leq \underbrace{\text{Loss}^{S,\rho}\left(\frac{\hat{h}}{\|\hat{h}\|_1}\right)}_{\text{Emp. Margin-Loss}} + \frac{1}{\sqrt{s}} \cdot \left( 4\sqrt{\frac{d \ln\left(\frac{\exp^1 s}{d}\right)}{\rho^2}} + \sqrt{\frac{\ln \frac{1}{\delta}}{2}} \right)$$

mit Wahrscheinlichkeit mindestens  $1 - \delta$ .

Beachte  $\text{Loss}_{0-1}(\hat{h}) \leq \text{Loss}_D(\hat{h})$ .

$\frac{1}{\sqrt{s}} \ll \rho$ , sonst ist die Schranke bedeutungslos.

Was bedeutet dieses Ergebnis für  $\hat{\mathcal{J}}$ ?

# Bagging



## Eingabe:

Ein Lernalgorithmus  $A$ , eine Menge  $S$  von  $s$  klassifizierten Beispielen sowie eine Klasse  $\mathcal{H}$  von Hypothesen mit Wertebereich  $\{-1, 1\}$ .

- (1) For  $i = 1$  to  $2k + 1$  do
  - (1a) Wähle  $s$  klassifizierte Beispiele (mit Zurücklegen) aus  $S$ .  
/\* Im Durchschnitt wird ein Anteil von  $1 - \exp^{-1} \approx 0.632$  aller Beispiele aus  $S$  gewählt. \*/
  - (1b)  $A$  berechne die Hypothese  $h_i \in \mathcal{H}$  für die ausgewählten Beispiele.
- (2) Gib die Mehrheitshypothese

$$h := \text{sign}\left(\sum_{i=1}^{2k+1} h_i\right)$$

aus.

# Bagging: Was sind die Erfahrungen?

- (a) Die Mehrheitshypothese gehört zur Hypothesenklasse  $\hat{\mathcal{H}}$ .
  - ▶ Ein *großer Margin* fast überall schließt *Overfitting* aus.
  
- (b) Bagging ist häufig für **instabile** Lernalgorithmen  $A$  erfolgreich.
  - ▶  $A$  ist instabil „ $\iff$ “  $A$  reagiert auf kleine Änderungen in der Beispielmenge mit stark modifizierten Hypothesen.
  - ▶ Algorithmen für Entscheidungsbäume und neuronale Netzwerke sind anscheinend Beispiele solcher instabiler Lernalgorithmen.

# AdaBoost

# Fehlerreduktion: Was geht und was geht nicht?

**Die Annahme:** Für jede Beispielmenge  $S$  gibt es eine konsistente Hypothese  $h \in \mathcal{H}$ .

Es gelte  $\theta > 0$ . Lässt sich eine konsistente Hypothese  $h \in \mathcal{H}$  **effizient** bestimmen, wenn ein **effizienter** Algorithmus  $A$

- ? für **jede** Verteilung hochwahrscheinlich Hypothesen  $g \in \mathcal{H}$  mit Fehler  $\leq \frac{1}{2} - \theta$  findet bzw
- ? für eine **bestimmte** Verteilung  $D$  hochwahrscheinlich Hypothesen  $g \in \mathcal{H}$  mit Fehler  $\leq \frac{1}{2} - \theta$  findet?

$A$  ist ein effizienter PAC-Algorithmus mit dem relativ großem Fehler

$$\varepsilon \leq \frac{1}{2} - \theta.$$

1. Fordere eine Beispielmenge  $S$  an.
2. Wenn  $A$  eine konsistente Hypothese  $h$  bestimmt: ✓
3. Sonst wiederhole  $T$ -mal
  - ▶ Erhöhe die „Wahrscheinlichkeit“ falsch klassifizierter Beispiele auf 50% und fordere  $A$  auf, eine neue Hypothese  $h$  zu liefern.
  - ▶ Halte, falls  $h$  konsistent ist.
4. Gib „so etwas“ wie eine Mehrheitshypothese aus.

# AdaBoost: Adaptive Boosting

**Eingabe:** Ein Lernalgorithmus  $A$  und eine Menge  $S = \{(x_i, y_i) : 1 \leq i \leq s\}$  von  $s$  klassifizierten Beispielen.

1.  $D_1$  ist die Gleichverteilung auf  $S$ . Setze  $w_i^{(1)} := 1$  für  $i = 1, \dots, s$ .  
/\* Das anfängliche Gesamtgewicht ist  $W_1 := s$ . \*/
2. For  $t = 1$  to  $T$  do /\* Der Parameter  $T$  ist zu bestimmen. \*/
  - ▶  $A$  bestimmt eine Hypothese  $h_t$  mit „Trainingsfehler“

$$\varepsilon_t = \text{prob}_{i \sim D_t} [h_t(x_i) \neq y_i].$$

- ▶ Setze  $\beta_t := \frac{\varepsilon_t}{1 - \varepsilon_t}$  und aktualisiere  $w_i^{(t+1)} := w_i^{(t)} \cdot \beta_t$  für alle durch  $h_t$  richtig klassifizierten Beispiele  $x_i$ .  
/\* Das Gesamtgewicht richtig (bzw. falsch) klassifizierter Beispiele ist  $(1 - \varepsilon_t) \cdot W_t \cdot \beta_t = \varepsilon_t \cdot W_t \implies$  Es ist  $W_{t+1} = 2\varepsilon_t W_t \implies W_{t+1} < W_t$ . \*/
  - ▶ Definiere die Verteilung  $D_{t+1}$  durch  $\text{prob}[x_i] := \frac{w_i}{\sum_{i=1}^s w_i}$ .
3. Setze  $\alpha_t := \ln\left(\frac{1}{\beta_t}\right)$  und gib die Hypothese  $\text{sign}\left(\sum_{t=1}^T \alpha_t h_t\right)$  aus.  
/\* Je kleiner der Fehler von  $h_t$ , umso größer das Gewicht  $\alpha_t$ . \*/

# Analyse von AdaBoost: Ein Überblick

Algorithmus  $A$  besitze auf jeder Verteilung  $D_t$  einen Fehler  $\varepsilon_t \leq \frac{1}{2} - \theta$ .

Die Analyse wird in zwei Schritten durchgeführt:

1. Zeige: Die Mehrheits-Hypothese  $\text{sign}(h)$  ist ab einer bestimmten Rundenzahl  $T$  **konsistent** mit den  $s$  Beispielen.
2. Aber

$$h \in \mathcal{H}_T := \left\{ \text{sign} \left( \sum_{i=1}^T \alpha_i h_i \right) \mid h_1, \dots, h_T \in \mathcal{H}, \alpha \geq 0 \right\}$$

und die VC-Dimension von  $\mathcal{H}_T$  **wächst** mit  $T \implies$

- ▶ u.U. sind  $S > s$  Beispiele für erfolgreiches Lernen in  $\mathcal{H}_T$  erforderlich.
- ▶ Aber dann **steigt** die Rundenzahl  $T$  für die größere Beispielzahl  $S$ .

**Catch-22?**

AdaBoost ist eine Variante von **Weighted Majority**:

- Die Beispiele  $x_i$  übernehmen die Rolle der Experten.
- Die Gewichtssetzung favorisiert informative Beispiele, also Beispiele, die häufig falsch klassifiziert werden.

Wir übernehmen Ideen aus der Analyse von Weighted Majority:  
Bestimme das Gesamtgewicht  $W_t$  zum Zeitpunkt  $t$ .

- ▶  $W_1 = s$ .
- ▶  $W_{t+1} = 2\varepsilon_t \cdot W_t$ .



Wann klassifiziert AdaBoost das Beispiel  $x_i$  falsch?

- $x_i$  werde zu Zeitpunkten in  $I_r \subseteq \{1, \dots, T\}$  richtig und
- zu Zeitpunkten in  $I_f \subseteq \{1, \dots, T\}$  falsch klassifiziert.

- Für  $\beta_t = \frac{\varepsilon_t}{1-\varepsilon_t}$  gilt

$$\sum_{t \in I_r} \ln\left(\frac{1}{\beta_t}\right) \leq \sum_{t \in I_f} \ln\left(\frac{1}{\beta_t}\right)$$

- Wir exponentieren beide Seiten

$$\prod_{t \in I_r} \frac{1}{\beta_t} = e^{\sum_{t \in I_r} \ln\left(\frac{1}{\beta_t}\right)} \leq e^{\sum_{t \in I_f} \ln\left(\frac{1}{\beta_t}\right)} = \prod_{t \in I_f} \frac{1}{\beta_t}.$$

- Wenn AdaBoost  $x_i$  falsch klassifiziert, dann ist

$$\prod_{t \in I_f} \beta_t \leq \prod_{t \in I_r} \beta_t \text{ bzw. } \prod_{t=1}^T \sqrt{\beta_t} \leq \prod_{t \in I_r} \beta_t.$$

Als Konsequenz einer falschen Klassifizierung folgt  $\prod_{t=1}^T \sqrt{\beta_t} \leq \prod_{t \in I_r} \beta_t$ .

- Das endgültige Gewicht  $w_i^{(T+1)}$  von  $x_i$  stimmt mit  $\prod_{t \in I_r} \beta_t$  überein,
  - ▶ da wir mit  $\beta_t$  multiplizieren, wenn Hypothese  $h_t$  richtig klassifiziert.
- Wenn AdaBoost das Beispiel  $x_i$  falsch klassifiziert, dann muss das endgültige Gewicht  $w_i^{(T+1)}$  groß sein, denn

$$w_i^{(T+1)} = \prod_{t \in I_r} \beta_t \geq \prod_{t=1}^T \sqrt{\beta_t}.$$

Wenn AdaBoost einen Fehler  $\mu$  auf der Trainingsmenge  $S$  hat, dann

$$\sum_{i=1}^S w_i^{(T+1)} \geq \sum_{i, \text{AdaBoost irrt auf } x_i} w_i^{(T+1)} \geq \mu S \cdot \prod_{t=1}^T \sqrt{\beta_t}$$

Je größer der Fehler  $\mu$  auf der Trainingsmenge, umso größer ist die Gewichtssumme  $W_{T+1}$  am Ende der Berechnung.

Aber wie groß kann die Gewichtssumme  $W_{T+1}$  werden?

(1) Wir haben bereits festgestellt, dass  $W_{t+1} = 2\varepsilon_t \cdot W_t$  gilt. Also folgt

$$W_{T+1} = W_1 \cdot \prod_{t=1}^T (2\varepsilon_t) = s \cdot \prod_{t=1}^T (2\varepsilon_t).$$

(2) Bei einem großen Fehler  $\mu$  auf der Trainingsmenge folgt:

$$\begin{aligned} \mu s \cdot \prod_{t=1}^T \sqrt{\beta_t} &\leq W_{T+1} = s \cdot \prod_{t=1}^T (2\varepsilon_t), \text{ bzw} \\ \mu &\leq \prod_{t=1}^T \frac{2\varepsilon_t}{\sqrt{\beta_t}} = \prod_{t=1}^T 2\sqrt{\varepsilon_t \cdot (1 - \varepsilon_t)}, \end{aligned}$$

$$\text{denn } \beta_t = \frac{\varepsilon_t}{1 - \varepsilon_t}.$$

# Der Trainingserfolg: Zusammenfassung

Wenn  $h_t$  den Fehler  $\varepsilon_t$  hat, dann hat AdaBoost den Trainingsfehler

$$\mu \leq 2^T \cdot \prod_{t=1}^T \sqrt{\varepsilon_t \cdot (1 - \varepsilon_t)}.$$

(1) Falls  $\varepsilon_t \leq \varepsilon = 1/2 - \theta$  für alle  $t$  mit  $1 \leq t \leq T$  folgt

$$\begin{aligned} \mu &\leq 2^T \cdot \prod_{t=1}^T \sqrt{\varepsilon_t \cdot (1 - \varepsilon_t)} \leq 2^T \cdot \sqrt{\varepsilon^T \cdot (1 - \varepsilon)^T} \\ &= 2^T \cdot [(1/2 - \theta) \cdot (1/2 + \theta)]^{T/2} = [1 - 4 \cdot \theta^2]^{T/2}. \end{aligned}$$

(2) Nach  $O(\log_2 s)$  Iterationen findet AdaBoost eine konsistente Hypothese in

$$\mathcal{H}_T = \left\{ \text{sign} \left( \sum_{i=1}^T \alpha_i h_i \right) \mid h_1, \dots, h_T \in \mathcal{H}, \alpha \geq 0 \right\}.$$

Aber die VC-Dimension von  $\mathcal{H}_T$  *wächst* mit  $T$ .

Man kann für  $d = VC(\mathcal{H})$  zeigen, dass

$$VC(\mathcal{H}_T) = \mathcal{O}(d \cdot T \ln T)$$

für eine geeignete Konstante  $C_2$  gilt.

- 1 Nach  $\mathcal{O}(\log_2 s)$  Iterationen wird eine konsistente Hypothese gefunden.
  - ▶ Beispielzahl verdoppelt sich  $\implies T = T + \mathcal{O}(1)$
  - ▶  $\implies$  Beispielzahl für erfolgreiches Lernen wächst sublinear.
- 2 Für eine geeignete Konstante  $C$  benötigt AdaBoost höchstens

$$\frac{C}{\varepsilon} \cdot \left( d \cdot \ln^2\left(\frac{d}{\varepsilon}\right) \cdot \ln \ln\left(\frac{d}{\varepsilon}\right) + \ln\left(\frac{1}{\delta}\right) \right)$$

Beispiele.

Ein *zweiter* Ansatz:

Für jede reelle Zahl  $\rho$  gilt

$$\begin{aligned} \text{Loss}^{S,\rho}(h) &\leq 2^T \cdot \prod_{t=1}^T \sqrt{\varepsilon_t^{1-\rho} (1 - \varepsilon_t)^{1+\rho}} \\ &\leq \left( \underbrace{\prod_{t=1}^T (1 - 2\theta)^{1-\rho} (1 + 2\theta)^{1+\rho}}_{<1} \right)^{T/2}. \end{aligned}$$

Wähle  $\rho = \sqrt{\frac{C}{s}}$  für eine entsprechend groß gewählte Konstante  $C$  und der erwartete Hinge-Loss ist klein. (Beachte  $\frac{1}{\sqrt{s}} \ll \rho$ .)

Die Hypothese von AdaBoost hat einen kleinen Margin-Loss!

(a) Der Trainingsfehler ist klein.

- ▶ Nur richtig klassifizierte Beispiele werden herabgesetzt  $\implies W_{T+1}$  ist groß, wenn der endgültige Trainingsfehler groß ist.
- ▶  $W_{t+1} = 2\varepsilon_t \cdot W_t$ .
  - ★ Falls  $\varepsilon_t \leq \frac{1}{2} - \theta \implies W_{T+1}$  ist klein.
  - ★ Insbesondere: Das Gesamtgewicht fällt geometrisch.

(b) Wenn es PAC-Algorithmen gibt, dann hat die AdaBoost-Hypothese einen beweisbar kleinen Margin-Loss.

In typischen Anwendungen liegen aber keine PAC-Algorithmen vor!

- (1) Wie Bagging erreicht AdaBoost häufig eine Verbesserung, wenn das Lernverfahren **instabil** ist, d.h. auf relativ kleine Änderungen der Beispielmenge mit stark veränderten Hypothesen reagiert.
- (2) Allerdings reagiert AdaBoost sehr sensibel auf verrauschte Beispiele, wenn also konsistente Hypothesen nicht existieren.
  - ▶ AdaBoost versucht, das von den Verteilungen  $D_t$  verteilte Gewicht auf diese nutzlosen Beispiele zu *konzentrieren*
  - ▶ und verschlechtert sogar die Lernleistung.



# Boosting für neuronale Netzwerke: Boosted LeNet4

**LeNet4** ist ein neuronales Netz zur Ziffernerkennung mit Fehler 1.1%.

Boosting wird für  $T = 3$  ausgeführt.

- (1) Das erste Netz ist LeNet 4.
- (2) Ein zweites Netz erhält zur Hälfte vom ersten Netz falsch klassifizierte und zur anderen Hälfte richtig klassifizierte Ziffern.
- (3) Das dritte Netz erhält als Trainingsmenge alle von den ersten beiden Netzen unterschiedlich klassifizierten Ziffern.
- (4) Das erste Netz ist bereits sehr gut:  
Vergrößere die Trainingsmenge durch zufällige Verzerrungen.
- (5) Die Hypothesen der drei Netze werden gleichgewichtet summiert.

Das „3-er Ensemble“ erreichte eine Fehlerrate von ungefähr 0.7%.

# Entscheidungsbäume

- Beispiele werden aus der Menge

$$X = \prod_{i=1}^n \Sigma_i$$

gewählt. Statt Komponenten spricht man von *Attributen*.

- ▶ Attribute sind *nominal* ( $|\Sigma_i| < \infty$ ) oder
- ▶ *numerisch* ( $\Sigma_i = \mathbb{N}, \mathbb{Z}, \mathbb{Q}, \mathbb{R}, \dots$ ).
- Die unbekannte Zielfunktion  $f$  klassifiziert Beispiele mit Werten aus einer endlichen Menge  $C$  von Klassifizierungen.
  - ▶ Möglicherweise sind die Werte einiger Attribute fehlerhaft, beziehungsweise fehlen in einigen Beispielen.
- Hypothesen werden durch *Entscheidungsbäume* repräsentiert:
  - ▶ Jedem Knoten  $v$  wird ein Attribut  $j \in \{1, \dots, n\}$  zugewiesen.
  - ▶  $v$  besitzt für jeden Attributwert  $\sigma \in \Sigma_j$  ein Kind.
  - ▶ Jedes Blatt wird mit einer Klassifikation  $c \in C$  markiert.

# Das Lernen von Entscheidungsbäumen, Algorithmus C4.5

$p = (p_1, \dots, p_k)$  sei eine Wahrscheinlichkeitsverteilung. Dann ist

$$H(p) = \sum_{i=1}^k -p_i \cdot \log p_i = \sum_{i=1}^k p_i \cdot \log_2\left(\frac{1}{p_i}\right)$$

die Entropie von  $p$ .

- Würfle mit Wahrscheinlichkeit  $p_i$  für Ergebnis  $i$ .
- $H(p)$  misst die *Unsicherheit* über das Ergebnis, bzw. den *mittleren Informationsgehalt* des Ergebnisses.
  - ▶ Die Unsicherheit ist maximal für die Gleichverteilung  $p_1 = \dots = p_k = \frac{1}{k}$ : Es ist  $H(p) = \log_2 k$ .
  - ▶ Die Unsicherheit ist minimal, wenn das Experiment deterministisch ist, wenn also  $p_i = 1$  für ein  $i$  gilt: Es ist  $H(p) = 0$ .

- (1) In der ersten Phase wird ein „Riesenbaum“ Knoten für Knoten, im Top-Down Ansatz gebaut. Wenn Knoten  $v$  hinzugefügt wird:
  - ▶ Wähle ein Attribut für  $v$ , das die Unsicherheit über die Klassifizierung der Beispiele, die  $v$  erreichen, *minimiert*.
  - ▶ Wenn Attribut  $j$  gewählt wird, dann erhält  $v$  für jeden Attributwert  $\sigma \in \Sigma_j$  ein Kind.

Benutze ein Greedy Kriterium: Einmal getroffene Zuweisungen werden nicht mehr rückgängig gemacht.

- (2) In der zweiten Phase wird der Riesenbaum komprimiert, um Overfitting zu verhindern.

Angenommen, wir haben den Knoten  $v$  erreicht.  
Welches der  $n$  Attribute sollten wir dem Knoten  $v$  zuweisen?

Sei  $S_v$  die Menge der klassifizierten Beispiele, die den Knoten  $v$  erreichen. Für Attribut  $j$  und Klassifikation  $a$  setze

$$S_v^j(a) = \{ x \in S_v \mid x_j = a \}.$$

- (1) Es sei  $p_c^j(a)$  die Wahrscheinlichkeit, dass ein Beispiel in  $S_v^j(a)$  mit  $c$  klassifiziert ist.
- (2) Wähle für  $v$  ein Attribut  $j$ , das

$$G(j) = \sum_a \frac{|S_v^j(a)|}{|S_v|} \cdot H(p_c^j(a) \mid c \in C)$$

*minimiert*: Je kleiner die Unsicherheit über die endgültige Klassifizierung, umso größer der (erhoffte) Fortschritt.

$X = X' \cup V$  mit der Valierungsmenge  $V$ .

- *Ansatz 1*: Entfernung von Teilbäumen.

- ▶ Bestimme den Knoten  $v$ , dessen Klassifikationsfehler auf  $V$  minimal ist, wenn  $v$  durch
  - ★ ein Blatt  $b_v$  oder
  - ★ durch einen Teilbaum ersetzt wird.

Die alte Mehrheitsklassifikation von  $v$  wird übernommen.

- ▶ Wiederhole dieses Verfahren bis der Klassifikationsfehler ansteigt bzw. verfolge den MDL-Ansatz (**Minimum Description Length**)
  - ★ Erlaube einen Tradeoff zwischen Trainingsfehler und Baumgröße.

- *Ansatz 2*: Kleinste Regelmengen.

- ▶ Jedes Blatt des Baums liefert eine Regel, deren Bedingung der Konjunktion der Attribut-Werte auf dem Weg zum Blatt und deren Folgerung der Markierung des Blatts entspricht.
- ▶ Für jede Regel lösche solange nacheinander Attribut-Werte in der Bedingung bis der Klassifizierungsfehler ansteigt.



- ❶ Falsche Klassifikationen in geringer Anzahl werden durch den probabilistischen Ansatz abgefangen.
- ❷ Fehlende Attributwerte: Weise den Attributwert zu, der
  - ▶ unter allen Beispielen in  $S_V$  am häufigsten vorkommt oder
  - ▶ der unter allen Beispielen in  $S_V$  mit gleicher Klassifikation am häufigsten vorkommt.

- (a) Das Einsatzgebiet von Entscheidungsbäumen besteht vor Allem im Lernen oder in der Modellierung von **Expertensystemen**.
  - ✓ Das Lernergebnis ist leicht verständlich.
- (b) Kleine Entscheidungsbäume sind **ausdrucksschwach**:
  - ⚡ Ein Entscheidungsbaum, der alle Eingaben aus  $\{0, 1\}^n$  mit ungerader Parität akzeptiert, muss  $2^n$  Knoten besitzen!
- (c) Entscheidungsbäume reagieren „sensibel“ auf kleine Änderungen.
  - ▶ Knoten niedriger Tiefe haben weitaus größeren Einfluss als Knoten großer Tiefe.
  - ▶ Boosting kann zu einer Steigerung der Lernleistung führen:
    - ★ Die verschiedenen von AdaBoost berechneten Verteilungen auf der Beispielmenge lassen sich problemlos in die erste Phase einbauen.