Kapitel ML: III

III. Entscheidungsbäume

- Repräsentation und Konstruktion
- Splitting
- Entscheidungsbaumalgorithmen
- Pruning

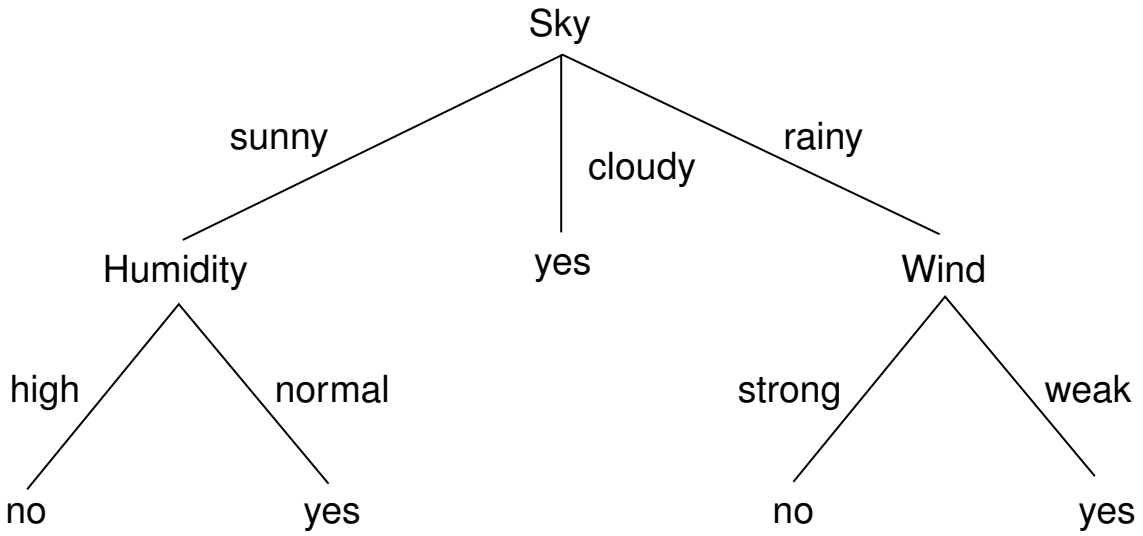
Repräsentation und Konstruktion

Spezifikation von Klassifikationsproblemen [vgl. ML:I-19]

- X sei ein Instanzenraum (Merkmalsraum) über endlich vielen Merkmalen.
- C sei eine Menge von Klassen.
- $c : X \rightarrow C$ sei der zu lernende Klassifikator für X.
- $D = \{(\mathbf{x}_1, c(\mathbf{x}_1)), \ldots, (\mathbf{x}_n, c(\mathbf{x}_n))\} \subseteq X \times C$ sei eine Menge von Lernbeispielen.

Repräsentation und Konstruktion

Entscheidungsbaum für das Konzept *EnjoySport*



- Nicht-Blattknoten repräsentieren Merkmale, die ausgewertet werden.
- Kanten repräsentieren Merkmalsausprägungen.
- Blätter definieren Klassen- bzw. Konzeptzuordnungen.

Aufteilung von X am Wurzelknoten:

$$X = \{\mathbf{x} \in X \mid \text{Sky}(\mathbf{x}) = \text{sunny}\} \cup \{\mathbf{x} \in X \mid \text{Sky}(\mathbf{x}) = \text{cloudy}\} \cup \{\mathbf{x} \in X \mid \text{Sky}(\mathbf{x}) = \text{rainy}\}$$

Bemerkungen:

- Eine Aufteilung wie im Beispiel ist nur bei endlichen Wertebereichen der Merkmale möglich, da für jede mögliche Ausprägung eine Kante zu einem Kindknoten führen muss.
- Jeder Pfad von dem Wurzelknoten zu einem Blattknoten entspricht einem Test hinsichtlich der Konjunktion der Merkmalsausprägungen, die mit diesem Pfad assoziiert sind. Dieser Test kann in Form einer Regel formuliert werden – Beispiel:

  IF Sky=rainy AND Wind=weak THEN EnjoySport=yes

- Entscheidungsbäume wurden populär durch die Vorstellung des ID3-Algorithmus von J. R. Quinlan im Jahr 1986.
- *Jeder* Knoten eines Entscheidungsbaumes kann als Klassifikator aufgefasst werden, der allen Elementen des Instanzenraumes X die eine, ihm zugeordnete Klasse zuweist.
- Zusammen bilden die Knoten eines Entscheidungsbaumes einen komplexen, typischerweise nicht-linearen Klassifikator.

Repräsentation und Konstruktion

Entscheidungsbaum (allgemein)

- Ein Entscheidungsbaum T ist ein endlicher Baum mit einem ausgezeichneten Wurzelknoten.
- Dem Wurzelknoten ist der gesamte Instanzenraum X zugeordnet. Jeder Nicht-Blattknoten t teilt eine Menge $X' \subseteq X$ in disjunkte Teilmengen, die den Kindknoten von t zugeordnet sind.
- Jedem Blattknoten ist eine Klasse aus C zugeordnet. *leaves*(T) bezeichne Menge aller Blattknoten in T.

- T klassifiziert ein Element $\mathbf{x} \in X$ durch eine rekursive Auswertung beginnend mit dem Wurzelknoten.
- Für Nicht-Blattknoten wird anhand eines Split-Kriteriums entschieden, zu welcher der entstehenden Teilmengen $\mathbf{x}$ gehört.
- Die einem Blattknoten zugeordnete Klasse wird als Klasse für $\mathbf{x}$ gewählt.
- Ein Entscheidungsbaum repräsentiert eine stückweise konstante Funktion.

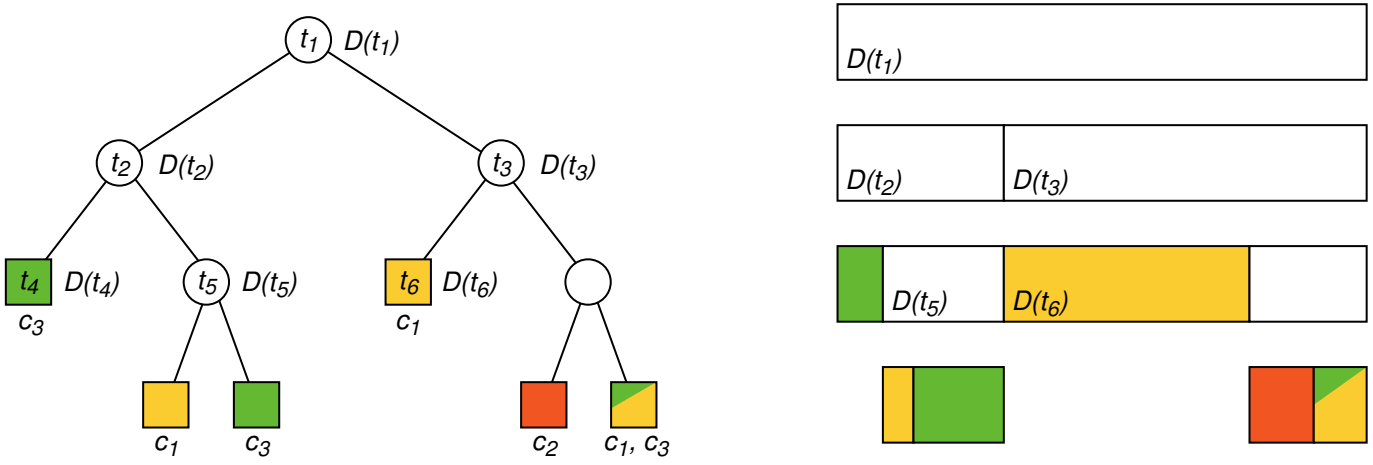
Repräsentation und Konstruktion

Entscheidungsbaum (allgemein)

Sei T ein Entscheidungsbaum, der das Konzept $c : X \rightarrow C$ approximiert, und sei t ein Knoten in T. Dann sei folgende Schreibweise vereinbart:

- $X(t)$ bezeichne die von Knoten t repräsentierte Teilmenge des Instanzenraumes X.
- $D(t)$ bezeichne die von Knoten t repräsentierte Teilmenge der Lernbeispiele D. Es gilt: $D(t) = \{(\mathbf{x}, c(\mathbf{x})) \in D \mid \mathbf{x} \in X(t)\}$

Illustration:



Repräsentation und Konstruktion

Algorithmenschema: Entscheidungsbaumkonstruktion

Algorithm:	*TDIDT*	Top Down Induction of Decision Tree
Input:	D	(Teil-)Menge von Lernbeispielen.
Output:	t	Wurzelknoten eines (Teil-)Entscheidungsbaumes.

TDIDT(D)

1. $t =$ *newNode*$()$
   class$(t) =$ *representativeClass*(D)
2. **IF** *pure*(D)
   THEN *return*(t)
   ELSE *criterion* $=$ *splitCriterion*(D)
3. $\{D_1, \ldots, D_s\} =$ *partition*$(D,$ *criterion*$)$
4. **FOREACH** D' **IN** $\{D_1, \ldots, D_s\}$ **DO**
   addSuccessor$(t,$ *TDIDT*$(D'))$
   ENDDO
5. *return*(t)

Repräsentation und Konstruktion

Algorithmenschema: Beispielklassifikation

Algorithm:	*CDT*	Classification by Decision Tree
Input:	$\mathbf{x}$	Merkmalsvektor.
	t	Wurzelknoten eines Entscheidungsbaumes T.
Output:	c	Klasse für Merkmalsvektor $\mathbf{x}$ bei Entscheidungsbaumes T.

CDT$(\mathbf{x}, t)$

```
1. IF isLeafNode(t)
   THEN return(class(t))
   ELSE return(CDT(x, splitSuccessor(t, x))
```

Bemerkungen:

- *representativeClass*(D)
  Bestimmt eine repräsentative Klasse für die Lernbeispiele D . Beachte, dass jeder Knoten durch späteres Pruning zu einem Blattknoten werden kann.
- *pure*(D)
  Entscheidet über eine weitere Aufteilung der Lernbeispiele D anhand der Purity in D.
- *splitCriterion*(D)
  Bestimmt das Split-Kriterium für die Lernbeispiele D.
- *partition*$(D, \textit{criterion})$
  Teilt die Lernbeispiele D entsprechend *criterion* in disjunkte Teilmengen.
- *addSuccessor*(t, t')
  Fügt einen Nachfolger t' für den Knoten t ein.

- *isLeafNode*(t)
  Bestimmt, ob der Knoten t ein Blattknoten ist.
- *splitSuccessor*$(t, \mathbf{x})$
  Gibt den Nachfolgerknoten von t an, der die Teilmenge von X repräsentiert, in der $\mathbf{x}$ liegt.

Repräsentation und Konstruktion

Einsatz von Entscheidungsbäumen

Charakteristika, die für den Einsatz von Entscheidungsbäumen sprechen:

- die Beispiele sind durch Attribut-Wert-Kombinationen beschreibbar
- die Zielfunktion ist diskretwertig (zunächst Bild- als auch Urbildbereich)
- Hypothesen sind grundsätzlich als Disjunktion aufgebaut
- die Trainingsmenge ist eventuell verrauscht

Anwendungsgebiete:

- Diagnose in der Medizin
- Fehlersuche in technischen Systemen
- Risikoanalyse bei der Kreditvergabe
- einfache Scheduling-Aufgaben, z. B. in der Terminplanung
- Klassifikation von Design-Schwächen im Softwareentwurf

Repräsentation und Konstruktion

Einsatz von Entscheidungsbäumen (Fortsetzung)

- Wie konstruiert man einen Entscheidungsbaum für D?
- Was ist ein guter Entscheidungsbaum?
- Wie soll entschieden werden, ob ein Knoten Blattknoten wird?
- Welche Klasse soll einem Blattknoten bei Nicht-Eindeutigkeit zugewiesen werden?

Repräsentation und Konstruktion

Güte von Entscheidungsbäumen

- Klassifikationsfehler.

- Größe.

Repräsentation und Konstruktion

Güte von Entscheidungsbäumen

- Klassifikationsfehler.

  Quantifiziert die Eindeutigkeit, mit der für ein $\mathbf{x}$ in einem Blattknoten von T auf Basis von D eine Entscheidung für eine Klasse erfolgt.

  Ein Entscheidungsbaum, dessen Blätter jeweils ein Beispiel aus D repräsentieren, macht keinen Klassifikationsfehler auf D.

- Größe.

  Von mehreren Theorien, die die gleichen Sachverhalte erklären, ist die einfachste allen anderen vorzuziehen *(Occam's Razor)*:

  Entia non sunt multiplicanda praeter necessitatem oder *sine necessitate.*

  [Johannes Clauberg (1622-1665)]

  Hier: unter allen Entscheidungsbäumen mit minimalem Klassifikationsfehler wählen wir den mit minimaler Größe.

Repräsentation und Konstruktion

Güte von Entscheidungsbäumen: Größe

- Blattanzahl
- Baumhöhe
- externe Pfadlänge
- gewichtete externe Pfadlänge

Repräsentation und Konstruktion

Güte von Entscheidungsbäumen: Größe

- Blattanzahl

  Die Blattanzahl entspricht der Anzahl der Regeln, die man aus einem Entscheidungsbaum generiert.

- Baumhöhe

  Die Baumhöhe entspricht der maximalen Regellänge, d. h. der Anzahl der maximal zu verifizierenden Prämissen für eine Entscheidung.

- externe Pfadlänge

  Die externe Pfadlänge ist definiert als die Summe der Längen aller Pfade von dem Wurzelknoten zu einem Blatt. Sie entpricht dem Speicherplatz für eine aus dem Baum generierte Regelmenge.

- gewichtete externe Pfadlänge

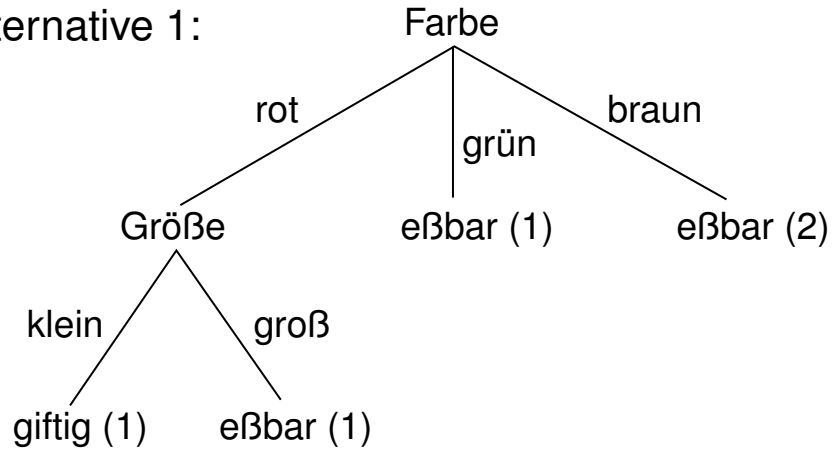
  Die gewichtete externe Pfadlänge wird berechnet wie die externe Pfadlänge, wobei die Längen der Pfade auf Basis der Anzahl der assoziierten Fallbeispiele in D gewichtet werden.

Repräsentation und Konstruktion

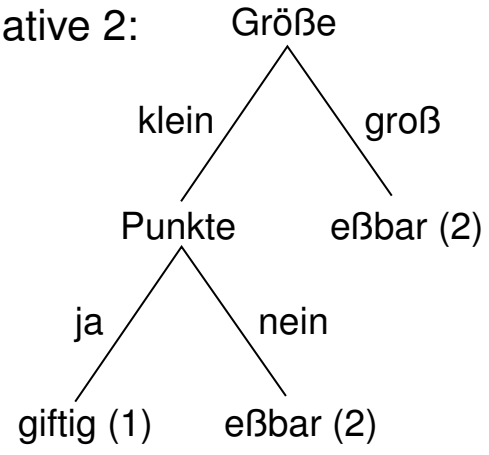
Güte von Entscheidungsbäumen: Größe

Folgende Alternativen klassifizieren alle Beispiele aus D korrekt:

Alternative 1:



Alternative 2:



Kriterium	Alternative 1	Alternative 2
Blattanzahl	4	3
Baumhöhe	2	2
externe Pfadlänge	6	5
gewichtete externe Pfadlänge	7	8

Repräsentation und Konstruktion

Güte von Entscheidungsbäumen: Größe

Satz 1 (externe Pfadlänge $< k$)
Das Problem, ob für eine Trainingsmenge D ein Entscheidungsbaum mit einer durch k beschränkten externen Pfadlänge existiert, ist NP-vollständig.

Repräsentation und Konstruktion

Güte von Entscheidungsbäumen: Klassifikationsfehler

In einem Entscheidungsbaum T für die Beispielmenge D wird einem Knoten t, der die Menge $D(t) \subseteq D$ repräsentiert, die Klasse *class*(t) auf Basis folgender Überlegung zugewiesen:

$$\textit{class}(t) = \operatorname*{argmax}_{c \in C} \frac{|\{(\mathbf{x}, c(\mathbf{x})) \in D(t) : c(\mathbf{x}) = c\}|}{|D(t)|}$$

Repräsentation und Konstruktion

Güte von Entscheidungsbäumen: Klassifikationsfehler

In einem Entscheidungsbaum T für die Beispielmenge D wird einem Knoten t, der die Menge $D(t) \subseteq D$ repräsentiert, die Klasse *class*(t) auf Basis folgender Überlegung zugewiesen:

$$\textit{class}(t) = \operatorname*{argmax}_{c \in C} \frac{|\{(\mathbf{x}, c(\mathbf{x})) \in D(t) : c(\mathbf{x}) = c\}|}{|D(t)|}$$

Schätzer für die Missklassifikationswahrscheinlichkeit eines Beispiels im Knoten t:

$$\textit{Err}(D(t)) = 1 - \max_{c \in C} \frac{|\{(\mathbf{x}, c(\mathbf{x})) \in D(t) : c(\mathbf{x}) = c\}|}{|D(t)|}$$

Trainingsfehler *Err*(D, T) für den Entscheidungsbaum T:

$$\textit{Err}(D, T) = \sum_{t \in \textit{leaves}(T)} \textit{Err}(D(t)) \cdot \frac{|D(t)|}{|D|}$$

Bemerkungen:

- Für einen Entscheidungsbaum T kann die theoretische Missklassifikationsrate $\mathit{Err}^*(T)$ auf Grundlage eines Wahrscheinlichkeitsmaßes P über C (anstatt durch relative Häufigkeiten über D) angegeben werden. Für einen Knoten t in T wird die Wahrscheinlichkeit der Missklassifikation minimal, falls gilt:

$$\mathit{class}(t) = \operatorname*{argmax}_{c \in C} \; P(c \mid X(t))$$

- Die Berechnung des Trainingsfehlers $\mathit{Err}(D, T)$ für einen Entscheidungsbaum T ist nur sinnvoll, wenn die induktive Lernannahme *(Inductive Learning Hypothesis)* angenommen wird. Diese fordert, dass die Verteilung von C in dem Instanzenraum X der Verteilung von C in Menge D der Lernbeispiele entspricht.

Repräsentation und Konstruktion

Güte von Entscheidungsbäumen: Missklassifikationskosten

Verallgemeinerung: Missklassifikation verursacht unterschiedlich hohe Kosten.
Kostenmaß für die Missklassifikation:

$$\mathit{cost}(c \mid c') \begin{cases} \geq 0 & \text{falls } c \neq c' \\ = 0 & \text{sonst} \end{cases}$$

Repräsentation und Konstruktion

Güte von Entscheidungsbäumen: Missklassifikationskosten

In einem Entscheidungsbaum T für die Beispielmenge D wird einem Knoten t, der die Menge $D(t) \subseteq D$ repräsentiert, die Klasse *class*(t) auf Basis folgender Überlegung zugewiesen:

$$\textit{class}(t) = \operatorname*{argmin}_{c \in C} \sum_{c' \in C} \textit{cost}(c' \mid c) \cdot \frac{|\{(\mathbf{x}, c(\mathbf{x})) \in D(t) : c(\mathbf{x}) = c\}|}{|D(t)|}$$

Repräsentation und Konstruktion

Güte von Entscheidungsbäumen: Missklassifikationskosten

In einem Entscheidungsbaum T für die Beispielmenge D wird einem Knoten t, der die Menge $D(t) \subseteq D$ repräsentiert, die Klasse *class*(t) auf Basis folgender Überlegung zugewiesen:

$$\textit{class}(t) = \operatorname*{argmin}_{c \in C} \sum_{c' \in C} \textit{cost}(c' \mid c) \cdot \frac{|\{(\mathbf{x}, c(\mathbf{x})) \in D(t) : c(\mathbf{x}) = c\}|}{|D(t)|}$$

Schätzer für die Missklassifikationswahrscheinlichkeit eines Beispiels im Knoten t:

$$\textit{Err}(D(t)) = \min_{c \in C} \sum_{c' \in C} \textit{cost}(c' \mid c) \cdot \frac{|\{(\mathbf{x}, c(\mathbf{x})) \in D(t) : c(\mathbf{x}) = c\}|}{|D(t)|}$$

Missklassifikationskosten *Err*(D, T) für den Entscheidungsbaum T:

$$\textit{Err}(D, T) = \sum_{t \in \textit{leaves}(T)} \textit{Err}(D(t)) \cdot \frac{|D(t)|}{|D|}$$

Kapitel ML: III (Fortsetzung)

III. Entscheidungsbäume

- Repräsentation und Konstruktion
- Splitting
- Entscheidungsbaumalgorithmen
- Pruning

Splitting

Sei t ein Blattknoten in einem unfertigen Entscheidungsbaum. Weiterhin bezeichne $D(t)$ die von t repräsentierten Lernbeispiele.

Überlegungen hinsichtlich einer weiteren Aufteilung von $D(t)$:

- Größe von $D(t)$.
- Reinheit von $D(t)$.
- Occam's Razor.

Splitting

Sei t ein Blattknoten in einem unfertigen Entscheidungsbaum. Weiterhin bezeichne $D(t)$ die von t repräsentierten Lernbeispiele.

Überlegungen hinsichtlich einer weiteren Aufteilung von $D(t)$:

- Größe von $D(t)$.
  $D(t)$ wird nicht weiter aufgeteilt, wenn die Anzahl der Beispiele $|D(t)|$ zu gering ist.
- Reinheit von $D(t)$.
  $D(t)$ wird nicht weiter aufgeteilt, wenn alle Beispiele in $D(t)$ zu derselben Klasse gehören.
- Occam's Razor.
  $D(t)$ wird nicht weiter aufgeteilt, wenn durch keinen Split für $D(t)$ der Entscheidungsbaum wesentlich verbessert wird.

Splitting

Definition 1 (Split)

Sei ein X Instanzenraums und $D = \{(\mathbf{x}, c(\mathbf{x})) \in D \mid \mathbf{x} \in X\}$ eine Menge von Lernbeispielen. Ein Split von X bzw. D ist eine Aufteilung von X bzw. D in disjunkte Teilmengen $X_1, \ldots, X_s$ bzw. $D_1, \ldots, D_s$. Ein Split, der eine Aufteilung in genau zwei disjunkte Teilmengen bewirkt, heißt *binärer Split*.

Splitting

Definition 1 (Split)
Sei ein X Instanzenraums und $D = \{(\mathbf{x}, c(\mathbf{x})) \in D \mid \mathbf{x} \in X\}$ eine Menge von Lernbeispielen. Ein Split von X bzw. D ist eine Aufteilung von X bzw. D in disjunkte Teilmengen $X_1, \ldots, X_s$ bzw. $D_1, \ldots, D_s$. Ein Split, der eine Aufteilung in genau zwei disjunkte Teilmengen bewirkt, heißt *binärer Split*.

Splitting ist abhängig vom Skalenniveau der Merkmale im Instanzenraum X:

1. Merkmal-Split für Merkmale A mit endlichem Wertebereich:
   $A = \{a_1, \ldots, a_k\}: \quad X = \{\mathbf{x} \in X : \mathbf{x}|_A = a_1\} \cup \ldots \cup \{\mathbf{x} \in X : \mathbf{x}|_A = a_k\}$

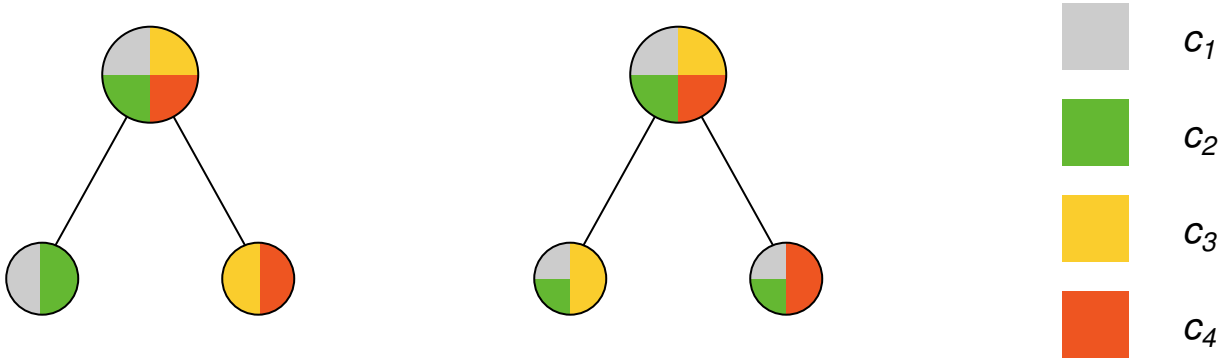
2. Standard-Split für nominalskalierte Merkmale A:
   $A' \subset A: \quad X = \{\mathbf{x} \in X : \mathbf{x}|_A \in A'\} \cup \{\mathbf{x} \in X : \mathbf{x}|_A \notin A'\}$

3. Standard-Split für ordinalskalierte (z.B. reellwertige) Merkmale A:
   $v \in \mathit{dom}(A): \quad X = \{\mathbf{x} \in X : \mathbf{x}|_A \geq v\} \cup \{\mathbf{x} \in X : \mathbf{x}|_A < v\}$

Splitting

Impurity

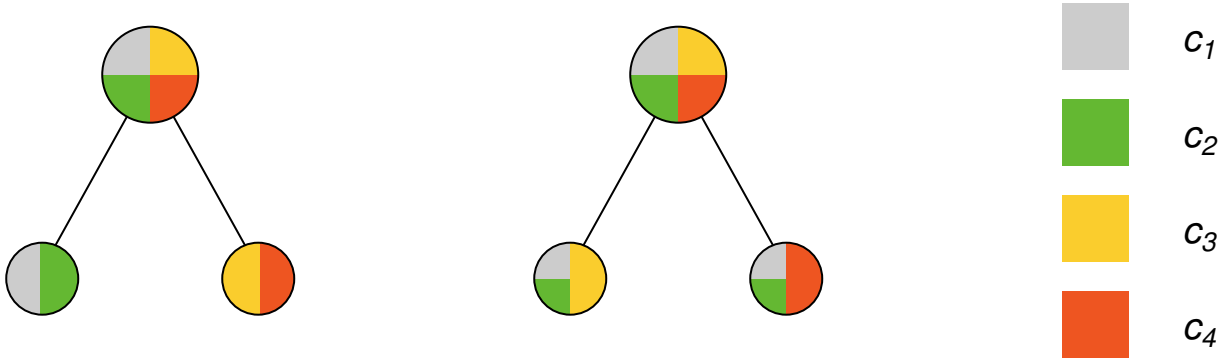
Sei $D(t)$ eine Menge von Lernbeispielen, in der $X(t)$ auf die Klassen $C = \{c_1, c_2, c_3, c_4\}$ verteilt ist. Illustration von zwei möglichen Splits:



Splitting

Impurity

Sei $D(t)$ eine Menge von Lernbeispielen, in der $X(t)$ auf die Klassen $C = \{c_1, c_2, c_3, c_4\}$ verteilt ist. Illustration von zwei möglichen Splits:



Der linke Split ist besser: ein guter Split minimiert die *Impurity* (Unreinheit) der entstehenden Teilmengen. Die Impurity ist eine Funktion auf den Teilmengen einer Lernmenge D.

Splitting

Impurity (Fortsetzung)

Definition 2 (Impurity-Funktion ι)

Für $s \in \mathbf{N}$ ist eine Impurity-Funktion $\iota : [0,1]^s \to \mathbf{R}$ eine (partielle) Funktion, die für alle Tupel $(p_1, \ldots, p_s)$ mit $p_i \geq 0$, $\sum_{i=1}^{s} p_i = 1$ definiert ist und folgende Eigenschaften hat:

(a) ι ist in den Punkten $(1, 0, \ldots, 0), \ldots, (0, \ldots, 0, 1)$ minimal.

(b) ι ist symmetrisch in allen Argumenten $p_1, \ldots, p_s$.

(c) ι ist im Punkt $(1/s, \ldots, 1/s)$ maximal.

Splitting

Impurity (Fortsetzung)

Definition 2 (Impurity-Funktion ι)
Für $s \in \mathbf{N}$ ist eine Impurity-Funktion $\iota : [0,1]^s \to \mathbf{R}$ eine (partielle) Funktion, die für alle Tupel $(p_1, \ldots, p_s)$ mit $p_i \geq 0$, $\sum_{i=1}^{s} p_i = 1$ definiert ist und folgende Eigenschaften hat:

(a) ι ist in den Punkten $(1, 0, \ldots, 0), \ldots, (0, \ldots, 0, 1)$ minimal.

(b) ι ist symmetrisch in allen Argumenten $p_1, \ldots, p_s$.

(c) ι ist im Punkt $(1/s, \ldots, 1/s)$ maximal.

Definition 3 (Impurity einer Menge von Lernbeispielen)
Sei D eine Menge von Lernbeispielen, $C = \{c_1, \ldots, c_k\}$ eine Menge von Klassen und $c : X \to C$ ein Konzept. Weiter sei $\iota : [0,1]^k \to \mathbf{R}$ eine Impurity-Funktion. Dann ist die Impurity von D, $\iota(D)$, wie folgt definiert:

$$\iota(D) = \iota\left(\frac{|\{(\mathbf{x}, c(\mathbf{x})) \in D : c(\mathbf{x}) = c_1\}|}{|D|}, \ldots, \frac{|\{(\mathbf{x}, c(\mathbf{x})) \in D : c(\mathbf{x}) = c_k\}|}{|D|}\right)$$

Splitting

Impurity (Fortsetzung)

Definition 4 (Impurity eines Entscheidungsbaums)

Sei D eine Menge von Lernbeispielen und T ein Entscheidungsbaum mit den Blattknoten *leaves*(T). Dann ist die Impurity von T hinsichtlich D, $\iota(T, D)$, wie folgt definiert:

$$\iota(T, D) = \sum_{t \in \textit{leaves}(T)} \iota(D(t)) \cdot \frac{|D(t)|}{|D|}$$

Splitting

Impurity (Fortsetzung)

Definition 4 (Impurity eines Entscheidungsbaums)
Sei D eine Menge von Lernbeispielen und T ein Entscheidungsbaum mit den Blattknoten *leaves*(T). Dann ist die Impurity von T hinsichtlich D, $\iota(T, D)$, wie folgt definiert:

$$\iota(T, D) = \sum_{t \in \textit{leaves}(T)} \iota(D(t)) \cdot \frac{|D(t)|}{|D|}$$

Definition 5 (Abnahme der Impurity)
Sei D eine Menge, und sei $D_1, \ldots, D_s$ ein Split von D. Dann ist die Abnahme der Impurity durch den Split, $\Delta\iota(\{D_1, \ldots, D_s\}, D)$, wie folgt definiert:

$$\Delta\iota(\{D_1, \ldots, D_s\}, D) = \iota(D) - \left(\frac{|D_1)|}{|D|} \cdot \iota(D_1) + \ldots + \frac{|D_s)|}{|D|} \cdot \iota(D_s) \right)$$

Bemerkungen:

- Die Impurity $\iota(T, D)$ misst die Güte eines Entscheidungsbaumes T auf Basis einer Lernmenge D.
- Die Splits für die Nachfolger eines Knoten sind unabhängig voneinander: sie können in beliebiger Reihenfolge durchgeführt werden, ohne die Impurity des resultierenden Baumes zu verändern.
- Üblicherweise verwendet der Algorithmus *TDIDT* eine Greedy-Methode (lokale Optimierung) zur Minimierung der Impurity $\iota(T, D)$.
- Die hier definierten Eigenschaften von $\iota(T, D)$ implizieren eine näherungsweise Minimierung der gewichteten externen Pfadlänge von T hinsichtlich D.

Splitting

Impurity-Funktionen für den 2-Klassen-Fall

Impurity-Funktion auf Basis der Missklassifikationsrate:

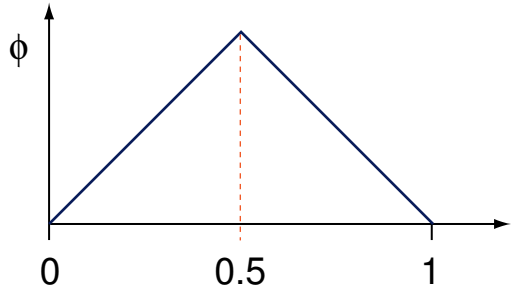
$$\iota_{misclass}(p_1, \ldots, p_s) = 1 - \max_i p_i$$

$$\iota_{misclass}(D(t)) = 1 - \max_{c \in C} \frac{|\{(\mathbf{x}, c(\mathbf{x})) \in D(t) : c(\mathbf{x}) = c\}|}{|D(t)|}$$

Im 2-Klassen-Fall gilt $p_2 = 1 - p_1$:

$$\iota_{misclass}(p_1, p_2) = 1 - \max\{p_1, p_2\} = \begin{cases} p_1 & \text{falls } 0 \leq p_1 \leq 0,5 \\ 1 - p_1 & \text{sonst} \end{cases}$$

$$\iota_{misclass}(D(t)) = 1 - \max\{\frac{|\{(\mathbf{x}, c(\mathbf{x})) \in D(t) : c(\mathbf{x}) = c_1\}|}{|D(t)|}, \frac{|\{(\mathbf{x}, c(\mathbf{x})) \in D(t) : c(\mathbf{x}) = c_2\}|}{|D(t)|}\}$$

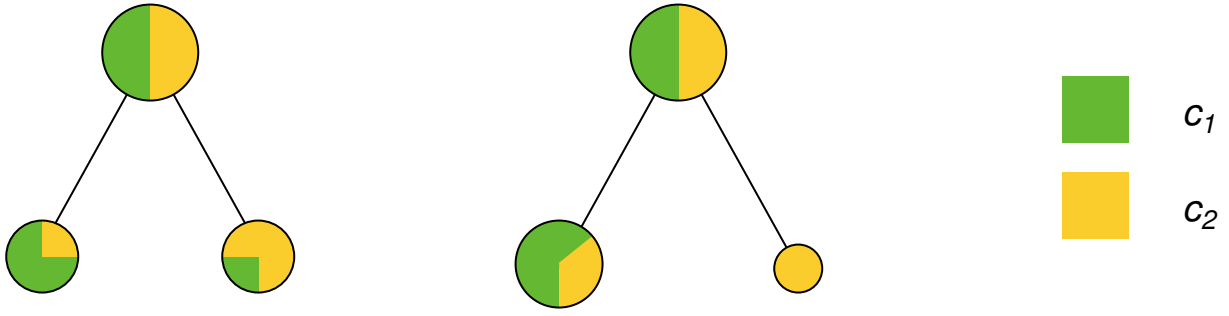


Splitting

Impurity-Funktionen für den 2-Klassen-Fall (Fortsetzung)

Probleme:

- $\Delta\iota = 0$ für alle Split-Kriterien möglich.
- Verwendung der Missklassifikationsrate als Impurity-Funktion kann zu schlechten Ergebnissen führen, denn pure Knoten werden nicht ausreichend bevorzugt:



Splitting

Impurity-Funktionen für den 2-Klassen-Fall (Fortsetzung)

Anforderungen an Impurity-Funktionen ι auf $[0, 1]$ gemäß Definition:

(a) $\iota(0, 1) = \iota(1, 0) = 0$ (Minimalität im puren Fall)

(b) $\iota(p_1, 1 - p_1) = \iota(1 - p_1, p_1)$ (Symmetrie)

(c) $\iota(0.5) = \max_{p_1 \in [0,1]} \iota(p_1, 1 - p_1)$

Splitting

Impurity-Funktionen für den 2-Klassen-Fall (Fortsetzung)

Anforderungen an Impurity-Funktionen ι auf $[0, 1]$ gemäß Definition:

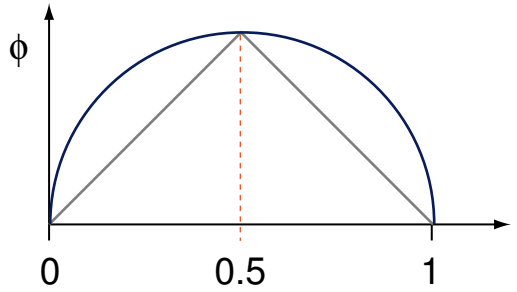
(a) $\iota(0, 1) = \iota(1, 0) = 0$ (Minimalität im puren Fall)

(b) $\iota(p_1, 1 - p_1) = \iota(1 - p_1, p_1)$ (Symmetrie)

(c) $\iota(0.5) = \max_{p_1 \in [0,1]} \iota(p_1, 1 - p_1)$

Änderung der Anforderung (c):

(c') ι zweimal stetig differenzierbar und $\iota''(p_1, 1 - p_1) < 0$ für $0 < p_1 < 1$.



Bemerkungen:

- Eine Funktion f mit Eigenschaft (c') ist streng konkav:

$$f(\lambda \mathbf{p} + (1-\lambda)\mathbf{p}') > \lambda f(\mathbf{p}) + (1-\lambda) f(\mathbf{p}')$$

  für $0 < \lambda < 1$ und $\mathbf{p} \neq \mathbf{p}'$. Beachte, dass $\mathbf{p}$ und $\mathbf{p}'$ Vektoren aus $[0,1]^k$ sind.

- Die durch die Missklassifikationsrate induzierte Impurity-Funktion ist konkav, aber nicht streng konkav.

Splitting

Impurity-Funktionen für den 2-Klassen-Fall (Fortsetzung)

Lemma 1

Für eine Impurity-Funktion ι mit den Eigenschaften (a), (b) und (c') folgt, dass für jeden Knoten t und jeden binären Split $\{D(t_L), D(t_R\}$ mit Nachfolgerknoten t_L und t_R gilt:

$$\Delta\iota(\{D(t_L), D(t_R\}, D(t)) \geq 0$$

Gleichheit gilt genau dann, wenn für $i = 1, 2$ gilt:

$$\frac{|\{(\mathbf{x}, c(\mathbf{x})) \in D(t) : c(\mathbf{x}) = c_i\}|}{|D(t)|} = \frac{|\{(\mathbf{x}, c(\mathbf{x})) \in D(t_L) : c(\mathbf{x}) = c_i\}|}{|D(t_L)|}$$

$$= \frac{|\{(\mathbf{x}, c(\mathbf{x})) \in D(t_R) : c(\mathbf{x}) = c_i\}|}{|D(t_R)|}$$

Bemerkungen:

- Wesentlicher Punkt im Beweis des Lemmas ist die Ausnutzung der Konkavheit von ι.
- Das Lemma lässt sich auf den allgemeinen Fall mit mehr als zwei Klassen und Splits mit einer Aufteilung in mehr als nur zwei Teilmengen übertragen.

Splitting

Impurity-Funktionen für den 2-Klassen-Fall (Fortsetzung)

Einfachstes Polynom mit Eigenschaften (a), (b) und (c'):

$$\iota(x) = a + bx + cx^2$$

Impurity-Funktion auf Basis des Gini-Index:

$$\iota_{Gini}(D(t)) = \frac{|\{(\mathbf{x}, c(\mathbf{x})) \in D(t) : c(\mathbf{x}) = c_1\}|}{|D(t)|} \cdot \frac{|\{(\mathbf{x}, c(\mathbf{x})) \in D(t) : c(\mathbf{x}) = c_2\}|}{|D(t)|}$$

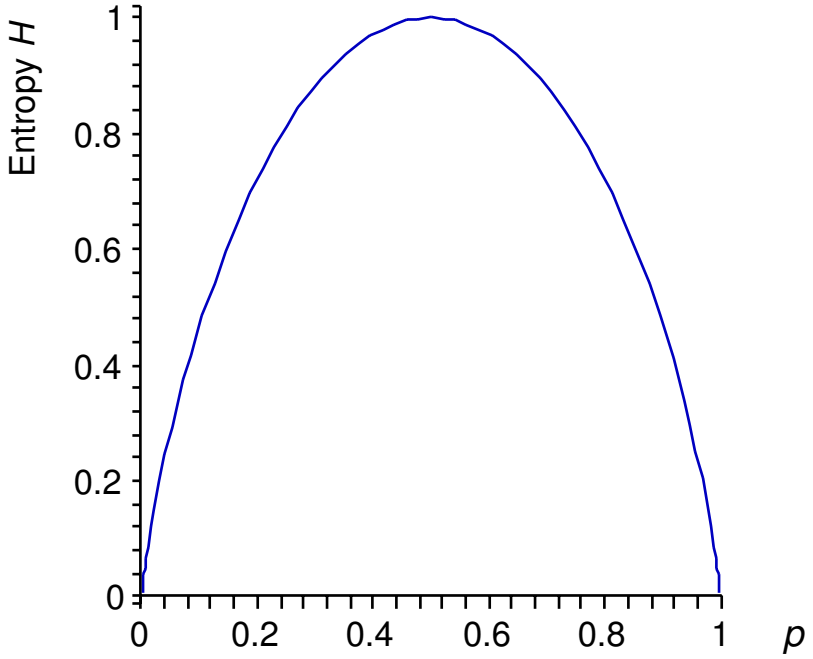
Impurity-Funktion auf Basis der Entropie:

$$\begin{aligned}\iota_{entropy}(D(t)) \;=\; & -\frac{|\{(\mathbf{x}, c(\mathbf{x})) \in D(t) : c(\mathbf{x}) = c_1\}|}{|D(t)|} \cdot \log \frac{|\{(\mathbf{x}, c(\mathbf{x})) \in D(t) : c(\mathbf{x}) = c_1\}|}{|D(t)|} \\ & -\frac{|\{(\mathbf{x}, c(\mathbf{x})) \in D(t) : c(\mathbf{x}) = c_2\}|}{|D(t)|} \cdot \log \frac{|\{(\mathbf{x}, c(\mathbf{x})) \in D(t) : c(\mathbf{x}) = c_2\}|}{|D(t)|}\end{aligned}$$

Splitting

Impurity-Funktionen für den 2-Klassen-Fall (Fortsetzung)

Illustration der Entropie:



$$\iota_{entropy}(p_{c_1}, p_{c_2}) = -p_{c_1}\log_2(p_{c_1}) - p_{c_2}\log_2(p_{c_2}), \quad \text{mit } p_{c_1} = p, \; p_{c_2} = 1 - p$$

Splitting

Impurity-Funktionen für den allgemeinen Fall: Gini-Index

$$\begin{aligned}\iota_{Gini}(D(t)) &= \left(\sum_{c\in C}\frac{|\{(\mathbf{x},c(\mathbf{x}))\in D(t):c(\mathbf{x})=c\}|}{|D(t)|}\right)^2 - \sum_{c\in C}\left(\frac{|\{(\mathbf{x},c(\mathbf{x}))\in D(t):c(\mathbf{x})=c\}|}{|D(t)|}\right)^2 \\ &= 1 - \sum_{c\in C}\left(\frac{|\{(\mathbf{x},c(\mathbf{x}))\in D(t):c(\mathbf{x})=c\}|}{|D(t)|}\right)^2\end{aligned}$$

Da $\iota_{Gini}(D(t))$ ein Polynom zweiten Grades ist, gilt weiter:

$$\Delta\iota_{Gini}(\{D(t_1),\ldots,D(t_s)\},D(t)) \geq 0$$

Gleichheit gilt genau dann, wenn für alle $c \in C$ gilt:

$$\begin{aligned}\frac{|\{(\mathbf{x},c(\mathbf{x}))\in D(t):c(\mathbf{x})=c\}|}{|D(t)|} &= \frac{|\{(\mathbf{x},c(\mathbf{x}))\in D(t_L):c(\mathbf{x})=c\}|}{|D(t_L)|} \\ &= \frac{|\{(\mathbf{x},c(\mathbf{x}))\in D(t_R):c(\mathbf{x})=c\}|}{|D(t_R)|}\end{aligned}$$

Splitting

Impurity-Funktionen für den allgemeinen Fall: Entropie

Definition 6 (Informationsgehalt, Entropie)
Tritt ein Ereignis A mit Wahrscheinlichkeit $P(A) > 0$ ein, dann ist der Informationsgehalt $I(A)$ dieses Eintretens definiert als:

$$I(A) = -\log_2(P(A))$$

Hat ein Versuch $\mathcal{A}$ die möglichen Ausgänge $A_1, \ldots, A_k$, so heißt der mittlere Informationsgehalt $H(\mathcal{A})$ die (Shannon-) Entropie des Versuchs $\mathcal{A}$:

$$H(\mathcal{A}) = -\sum_{i=1}^{k} P(A_i)\log_2(P(A_i))$$

Bemerkungen:

- Der Informationsgehalt eines Ereignisses ist um so größer, je seltener das Ereignis ist. Ein absolut sicheres Ereignis hat den Informationsgehalt 0.
- Die Entropie (mittlerer Informationsgehalt) gewichtet den Informationsgehalt für die möglichen Werte der Klassifikation mit den jeweiligen Wahrscheinlichkeiten.
- Es sei $0 \cdot \log_2(0) = 0$ vereinbart.

Splitting

Impurity-Funktionen für den allgemeinen Fall: Entropie (Fortsetzung)

Definition 7 (bedingte Entropie, Informationsgewinn)
Hat ein zweiter Versuch $\mathcal{B}$ die möglichen Ausgänge $B_1, \ldots, B_l$, dann ist die bedingte Entropie des kombinierten Versuchs $(\mathcal{A} \mid \mathcal{B})$ definiert als:

$$H(\mathcal{A} \mid \mathcal{B}) = \sum_{j=1}^{l} P(B_j) \cdot H(\mathcal{A} \mid B_j)$$

Der Informationsgewinn durch Versuch $\mathcal{B}$ ist definiert als $H(\mathcal{A}) - H(\mathcal{A} \mid \mathcal{B})$:

$$H(\mathcal{A}) - H(\mathcal{A} \mid \mathcal{B}) = H(\mathcal{A}) - \sum_{j=1}^{l} P(B_j) \cdot H(\mathcal{A} \mid B_j)$$

Bemerkungen:

- Der Informationsgewinn entspricht der erwarteten Verkleinerung der Entropie. In der deutschen Literatur wird teilweise die englische Bezeichnung *Information-Gain* für den Informationsgewinn verwendet.
- Im Kontext der Entscheidungsbäume entspricht der Versuch $\mathcal{A}$ der Klassifikation eines Beispiels $\mathbf{x}$ hinsichtlich des Zielkonzeptes. Mögliche Frage, um festzustellen welches Ereignis aus $\mathcal{A}$ eingetroffen ist: „Gehört $\mathbf{x}$ zur Klasse c?“ Der Versuch $\mathcal{B}$ entspricht der Auswertung eines Merkmals von $\mathbf{x}$. Mögliche Frage, um festzustellen welches Ereignis aus $\mathcal{B}$ eingetroffen ist: „Hat $\mathbf{x}$ hinsichtlich Merkmal B den Wert b?“
- Da es sich bei Zielkonzeptklassifikation und Merkmalauswertung oft um statistisch abhängige Ereignisse handelt, ist der Informationsgehalt der (für $\mathbf{x}$ noch unbekannten) Zielkonzeptklasse $c(\mathbf{x})$ nicht mehr so hoch, wenn man schon den Wert eines bestimmten Merkmals von $\mathbf{x}$ kennt. Der Informationsgewinn ist niemals negativ; im Falle statistisch unabhängiger Ereignisse ist er 0.
- Weil $H(\mathcal{A})$ konstant ist, entspricht die Bestimmung des Merkmals mit dem höchsten Informationsgehalt der Minimierung von $H(\mathcal{A} \mid \mathcal{B})$.
- Die ausgeschriebene Version von $H(\mathcal{A} \mid \mathcal{B})$ lautet:

$$H(\mathcal{A} \mid \mathcal{B}) = -\sum_{j=1}^{l} P(B_j) \cdot \sum_{i=1}^{k} P(A_i \mid B_j) \mathsf{log}_2(P(A_i \mid B_j))$$

Splitting

Impurity-Funktionen für den allgemeinen Fall: Entropie (Fortsetzung)

Die Formel $H(\mathcal{A}) - H(\mathcal{A} \mid \mathcal{B})$ für den Informationsgewinn entspricht der Abnahme der Impurity $\Delta\iota(\{D(t_1), \ldots, D(t_l)\}, D(t))$ im Knoten t eines Entscheidungsbaumes:

$$\Delta\iota = \iota(D(t)) - \left[\frac{|D(t_1)|}{|D(t)|} \cdot \iota(D(t_1)) + \ldots + \frac{|D(t_l)|}{|D(t)|} \cdot \iota(D(t_l))\right]$$

Zuordnung:

- $\mathcal{A}$ entspricht der Klassenzuordnung durch $c : X \to C$. A_i ist das Ereignis, dass ein $\mathbf{x} \in X(t)$ die Klasse c_i hat.
- $\mathcal{B}$ entspricht der Zuordnung von Werten $\{b_1, \ldots, b_l\}$ für ein Merkmal B mit endlichem Wertebereich. B_j ist das Ereignis, dass ein $\mathbf{x} \in X(t)$ bezüglich des Merkmals $\mathcal{B}$ den Wert b_j hat.

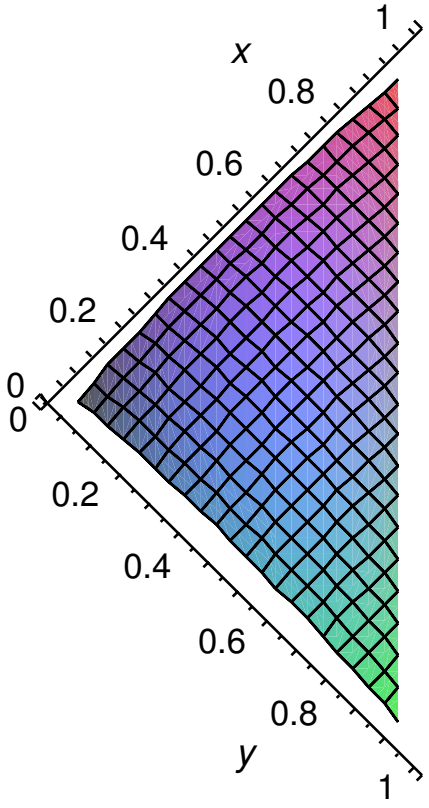
  $$X(t) = \{\mathbf{x} \in X(t) : \mathbf{x}|_B = b_1\} \cup \ldots \cup \{\mathbf{x} \in X(t) : \mathbf{x}|_B = b_l\}$$

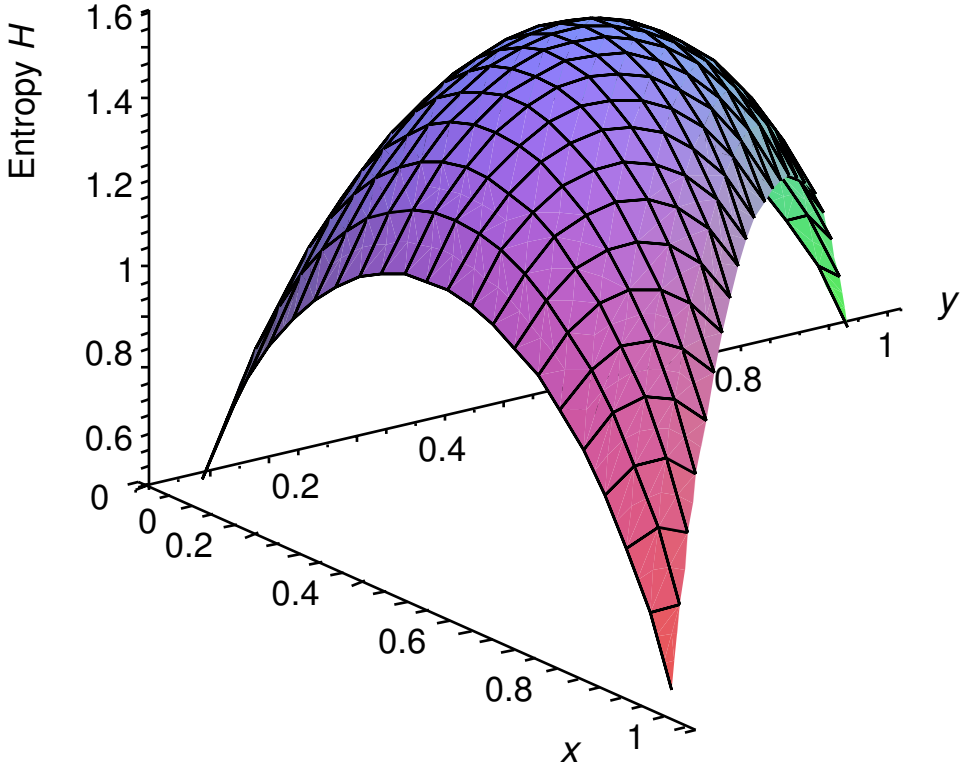
- $P(A_i), P(B_j), P(A_i \mid B_j)$ werden durch die relativen Häufigkeiten in $D(t)$ geschätzt.
- Verwendung der Impurity-Funktion $\iota : [0,1]^k \to \mathbf{R}$ mit $\iota(p_1, \ldots, p_k) = -\sum_{i=1,\ldots,|C|} p_i \cdot \log p_i$

Splitting

Impurity-Funktionen für den allgemeinen Fall: Entropie (Fortsetzung)

Illustration der Entropie:





Kapitel ML: III (Fortsetzung)

III. Classification and Regression Trees

- Repräsentation und Konstruktion
- Splitting
- Entscheidungsbaumalgorithmen
- Pruning

Entscheidungsbaumalgorithmen

Klassifikationsproblem (Wiederholung)

- X sei ein Instanzenraum (Merkmalsraum) über endlich vielen Merkmalen mit jeweils endlichem Grundbereich.
- C sei eine Menge von Klassen.
- $c : X \to C$ sei der zu lernende Klassifikator für X.
- $D = \{(\mathbf{x}_1, c(\mathbf{x}_1)), \ldots, (\mathbf{x}_n, c(\mathbf{x}_n))\} \subseteq X \times C$ sei eine Menge von Lernbeispielen.

Aufgabe: Konstruktion eines Entscheidungsbaums T für c auf Basis von D.

Entscheidungsbaumalgorithmen

Klassifikationsproblem (Wiederholung)

- X sei ein Instanzenraum (Merkmalsraum) über endlich vielen Merkmalen mit jeweils endlichem Grundbereich.
- C sei eine Menge von Klassen.
- $c : X \to C$ sei der zu lernende Klassifikator für X.
- $D = \{(\mathbf{x}_1, c(\mathbf{x}_1)), \ldots, (\mathbf{x}_n, c(\mathbf{x}_n))\} \subseteq X \times C$ sei eine Menge von Lernbeispielen.

Aufgabe: Konstruktion eines Entscheidungsbaums T für c auf Basis von D.

ID3-Algorithmus [Quinlan 1986]:

1. Als Splits sind nur vollständige Auswertungen bezüglich *eines* Merkmals erlaubt. Für Merkmal A mit Ausprägungen $\{a_1, \ldots, a_k\}$ gilt:
$$X = \{\mathbf{x} \in X : \mathbf{x}|_A = a_1\} \cup \ldots \cup \{\mathbf{x} \in X : \mathbf{x}|_A = a_k\}$$
2. Splits werden auf Basis des Informationsgewinns gemacht.

Entscheidungsbaumalgorithmen

ID3-Algorithmus [vgl. Mitchell 1997]

ID3(D, Attributes, Target)

- Create a node t for the tree.
- If all examples in D are positive, return the single-node tree t with label '+'.
- If all examples in D are negative, return the single-node tree t, with label '-'.
- If Attributes is empty, return the single-node tree t, with label of the most common value of Target in D.
- Otherwise:
  - Let A* be the attribute from Attributes that best classifies Examples.
  - The decision attribute of t is A*.
  - For each possible value a in A*:
    - Add a new tree branch below t, corresponding to the test A* = a.
    - Let D_a be the subset of D that have value a for A*.
    - If D_a is empty:
      Then below this new branch add a leaf node with label of the most common value of Target in D.
      Else below this new branch add the subtree ID3(D_a, Attributes – {A*}, Target).
- Return t.

Entscheidungsbaumalgorithmen

ID3-Algorithmus (formal)

ID3$(D, \mathit{Attributes}, \mathit{Target})$

1. $t = \mathit{createNode}()$
2. **IF** $\forall \langle \mathbf{x}, c(\mathbf{x}) \rangle \in D : c(\mathbf{x}) = 1$ **THEN** $\mathit{label}(t) =$ `'+'`, $\mathit{return}(t)$ **ENDIF**
3. **IF** $\forall \langle \mathbf{x}, c(\mathbf{x}) \rangle \in D : c(\mathbf{x}) = 0$ **THEN** $\mathit{label}(t) =$ `'-'`, $\mathit{return}(t)$ **ENDIF**
4. **IF** $\mathit{Attributes} = \emptyset$ **THEN** $\mathit{label}(t) = \mathit{mostCommonClass}(D, \mathit{Target})$, $\mathit{return}(t)$ **ENDIF**
5.
6.
7.
8.

Entscheidungsbaumalgorithmen

ID3-Algorithmus (formal)

ID3$(D, \mathit{Attributes}, \mathit{Target})$

1. $t = \mathit{createNode}()$
2. **IF** $\forall \langle \mathbf{x}, c(\mathbf{x}) \rangle \in D : c(\mathbf{x}) = 1$ **THEN** $\mathit{label}(t) =$ `'+'`, $\mathit{return}(t)$ **ENDIF**
3. **IF** $\forall \langle \mathbf{x}, c(\mathbf{x}) \rangle \in D : c(\mathbf{x}) = 0$ **THEN** $\mathit{label}(t) =$ `'-'`, $\mathit{return}(t)$ **ENDIF**
4. **IF** $\mathit{Attributes} = \emptyset$ **THEN** $\mathit{label}(t) = \mathit{mostCommonClass}(D, \mathit{Target})$, $\mathit{return}(t)$ **ENDIF**
5. $A^* = \text{argmax}_{A \in \mathit{Attributes}}(\mathit{informationGain}(D, A))$
6. $\mathit{label}(t) = A^*$
7.
8.

Entscheidungsbaumalgorithmen

ID3-Algorithmus (formal)

ID3$(D, \mathit{Attributes}, \mathit{Target})$

1. $t = \mathit{createNode}()$
2. **IF** $\forall\langle \mathbf{x}, c(\mathbf{x})\rangle \in D : c(\mathbf{x}) = 1$ **THEN** $\mathit{label}(t) =$ `'+'`, $\mathit{return}(t)$ **ENDIF**
3. **IF** $\forall\langle \mathbf{x}, c(\mathbf{x})\rangle \in D : c(\mathbf{x}) = 0$ **THEN** $\mathit{label}(t) =$ `'-'`, $\mathit{return}(t)$ **ENDIF**
4. **IF** $\mathit{Attributes} = \emptyset$ **THEN** $\mathit{label}(t) = \mathit{mostCommonClass}(D, \mathit{Target})$, $\mathit{return}(t)$ **ENDIF**
5. $A^* = \text{argmax}_{A \in \mathit{Attributes}}(\mathit{informationGain}(D, A))$
6. $\mathit{label}(t) = A^*$
7. **FOREACH** $a \in A^*$ **DO**

   $D_a = \mathit{selectExamples}(D, A^*, a)$

   IF $D_a = \emptyset$ **THEN**

   ELSE

   $\mathit{createEdge}(t, a, \mathit{ID3}(D_a, \mathit{Attributes} \setminus \{A^*\}, \mathit{Target}))$

   ENDIF

   ENDDO
8. $\mathit{return}(t)$

Entscheidungsbaumalgorithmen

ID3-Algorithmus (formal)

ID3$(D, \mathit{Attributes}, \mathit{Target})$

1. $t = \mathit{createNode}()$
2. **IF** $\forall\langle \mathbf{x}, c(\mathbf{x})\rangle \in D : c(\mathbf{x}) = 1$ **THEN** $\mathit{label}(t) =$ `'+'`, $\mathit{return}(t)$ **ENDIF**
3. **IF** $\forall\langle \mathbf{x}, c(\mathbf{x})\rangle \in D : c(\mathbf{x}) = 0$ **THEN** $\mathit{label}(t) =$ `'-'`, $\mathit{return}(t)$ **ENDIF**
4. **IF** $\mathit{Attributes} = \emptyset$ **THEN** $\mathit{label}(t) = \mathit{mostCommonClass}(D, \mathit{Target})$, $\mathit{return}(t)$ **ENDIF**
5. $A^* = \text{argmax}_{A \in \mathit{Attributes}}(\mathit{informationGain}(D, A))$
6. $\mathit{label}(t) = A^*$
7. **FOREACH** $a \in A^*$ **DO**

   $D_a = \mathit{selectExamples}(D, A^*, a)$

   IF $D_a = \emptyset$ **THEN**

   $t' = \mathit{createNode}()$

   $\mathit{label}(t') = \mathit{mostCommonClass}(D, \mathit{Target})$

   $\mathit{createEdge}(t, a, t')$

   ELSE

   $\mathit{createEdge}(t, a, \mathit{ID3}(D_a, \mathit{Attributes} \setminus \{A^*\}, \mathit{Target}))$

   ENDIF

   ENDDO
8. $\mathit{return}(t)$

Entscheidungsbaumalgorithmen

ID3-Algorithmus: Beispiel

Beschreibung von Pilzen:

	Farbe	Größe	Punkte	Verzehrbarkeit
1	rot	klein	ja	giftig
2	braun	klein	nein	essbar
3	braun	groß	ja	essbar
4	grün	klein	nein	essbar
5	rot	groß	nein	essbar

Anwendung des ID3-Algorithmus.

Entscheidungsbaumalgorithmen

ID3-Algorithmus: Beispiel (Fortsetzung)

1. Rekursionschritt, `Splitting` hinsichtlich des Merkmals „Farbe“:

$D|_{\text{Farbe}} =$

	giftig	essbar
rot	1	1
braun	0	2
grün	0	1

$\Rightarrow \quad |D_{\text{rot}}| = 2,\ |D_{\text{braun}}| = 2,\ |D_{\text{gruen}}| = 1$

Entscheidungsbaumalgorithmen

ID3-Algorithmus: Beispiel (Fortsetzung)

1. Rekursionschritt, Splitting hinsichtlich des Merkmals „Farbe“:

$D|_{\text{Farbe}} =$

	giftig	essbar
rot	1	1
braun	0	2
grün	0	1

$\rightarrow \quad |D_{\text{rot}}| = 2,\ |D_{\text{braun}}| = 2,\ |D_{\text{gruen}}| = 1$

Geschätzte a-Priori-Wahrscheinlichkeiten:

$$p_{\text{rot}} = \frac{2}{5} = 0.4, \quad p_{\text{braun}} = \frac{2}{5} = 0.4, \quad p_{\text{gruen}} = \frac{1}{5} = 0.2$$

Bedingte Entropie:

$$\begin{aligned} H(C \mid \text{Farbe}) &= -(0.4(\tfrac{1}{2}\log_2\tfrac{1}{2} + \tfrac{1}{2}\log_2\tfrac{1}{2}) + \\ &\quad 0.4(\tfrac{0}{2}\log_2\tfrac{0}{2} + \tfrac{2}{2}\log_2\tfrac{2}{2}) + \\ &\quad 0.2(\tfrac{0}{1}\log_2\tfrac{0}{1} + \tfrac{1}{1}\log_2\tfrac{1}{1})) = 0.4 \end{aligned}$$

$$\begin{aligned} H(C \mid \text{Groesse}) &\approx 0.46 \\ H(C \mid \text{Punkte}) &= 0.4 \end{aligned}$$

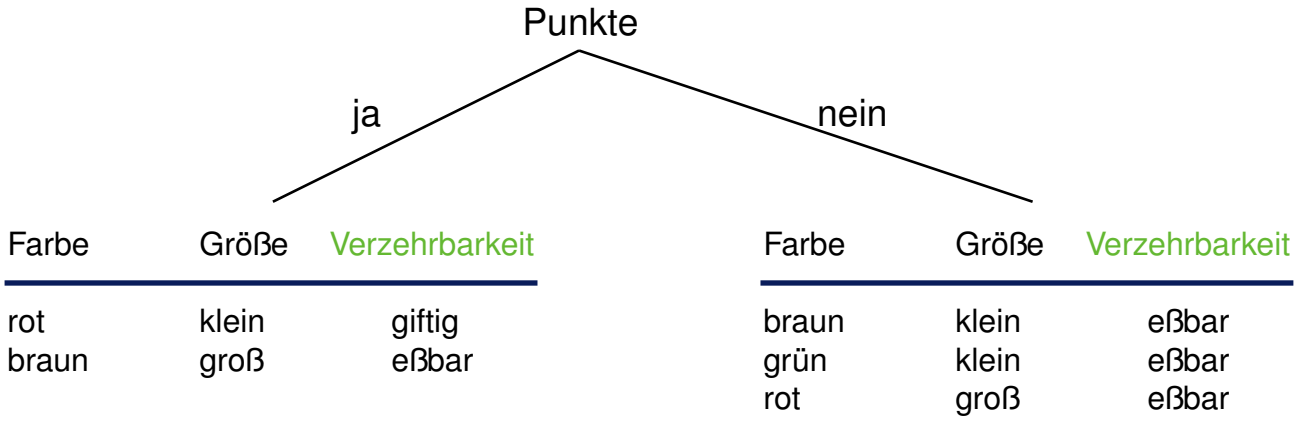
Bemerkungen:

- Der `Informationsgewinn` ist um so größer, je kleiner $H(C \mid \text{Merkmal})$ ist. Die Differenz $H(C) - H(C \mid \text{Merkmal})$ braucht nicht gebildet zu werden, weil $H(C)$ innerhalb einer Rekursionsstufe konstant ist.
- Im Beispiel ist in der ersten Rekursionsstufe der Informationsgewinn für die beiden Merkmale „Farbe“ und „Punkte“ maximal.

Entscheidungsbaumalgorithmen

ID3-Algorithmus: Beispiel (Fortsetzung)

Baum nach 1. Rekursionschritt:

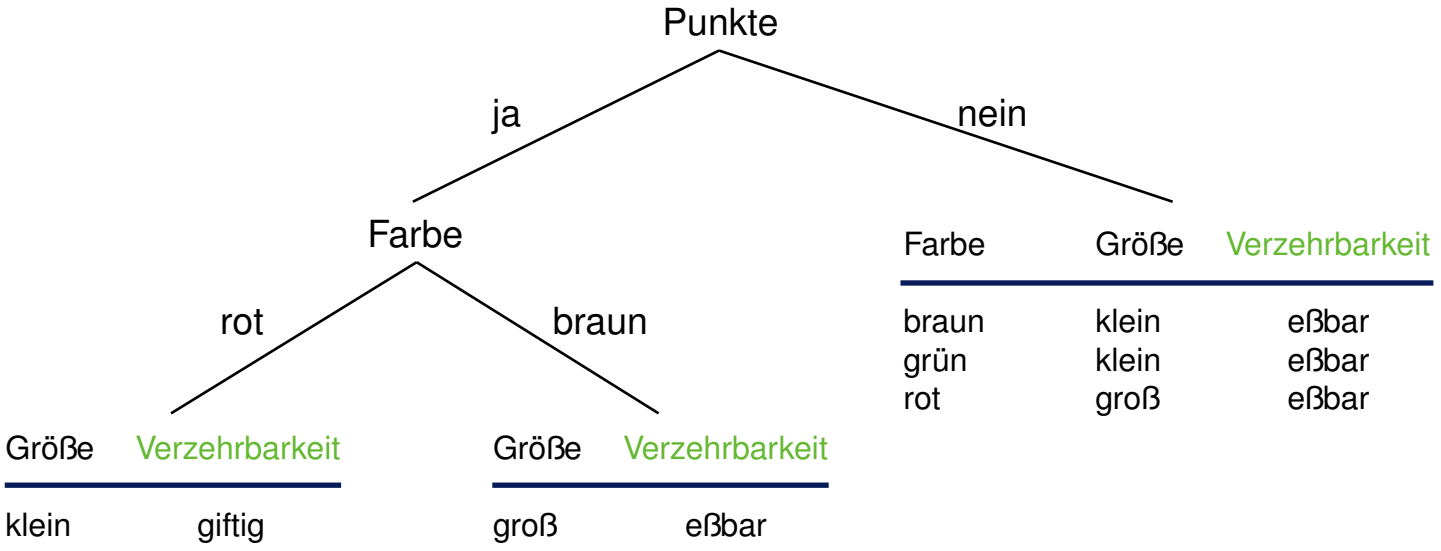


Das Merkmal „Punkte“ wurde ausgewählt (Schritt 5, ID3-Algorithmus).

Entscheidungsbaumalgorithmen

ID3-Algorithmus: Beispiel (Fortsetzung)

Baum nach 2. Rekursionschritt:

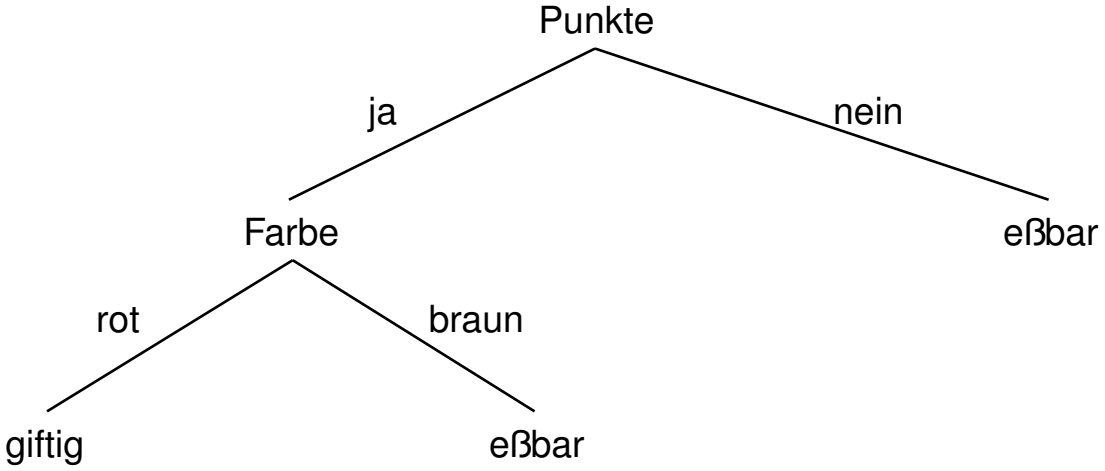


Das Merkmal „Farbe“ wurde ausgewählt (Schritt 5, ID3-Algorithmus).

Entscheidungsbaumalgorithmen

ID3-Algorithmus: Beispiel (Fortsetzung)

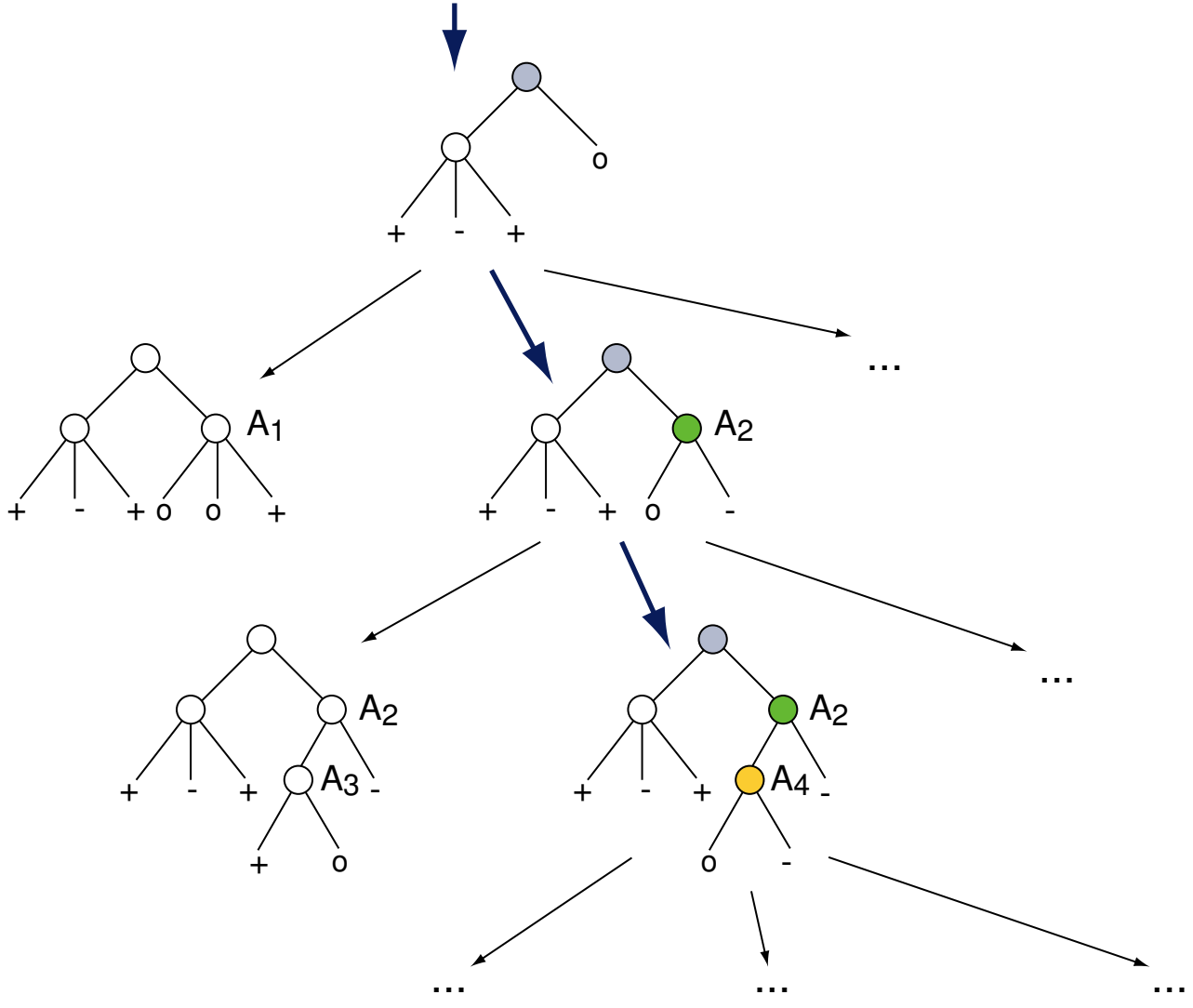
Baum nach 2. Rekursionschritt:



Endgültiger Entscheidungsbaum.

Entscheidungsbaumalgorithmen

Hypothesenraum von ID3



Entscheidungsbaumalgorithmen

Induktiver Bias von ID3

Der induktive Bias eines Algorithmus ist sein Prinzip, aus der Klassifikation von Trainingsbeispielen auf die Klassifikation neuer Beispiele zu schließen.

Beobachtungen:

- Entscheidungsbaumsuche findet im Raum *aller* Hypothesen statt.

  ➜

- Der ID3-Algorithmus trifft nur soviel Entscheidungen wie es Merkmale gibt.

  ➜

  ➜

Entscheidungsbaumalgorithmen

Induktiver Bias von ID3

Der induktive Bias eines Algorithmus ist sein Prinzip, aus der Klassifikation von Trainingsbeispielen auf die Klassifikation neuer Beispiele zu schließen.

Beobachtungen:

- Entscheidungsbaumsuche findet im Raum *aller* Hypothesen statt.
  - ➜ Zielkonzept muss im Hypothesenraum sein
- Der ID3-Algorithmus trifft nur soviel Entscheidungen wie es Merkmale gibt.
  - ➜ kein Backtracking
  - ➜ *lokale* Optimierung von Entscheidungsbäumen

Entscheidungsbaumalgorithmen

Induktiver Bias von ID3

Der induktive Bias eines Algorithmus ist sein Prinzip, aus der Klassifikation von Trainingsbeispielen auf die Klassifikation neuer Beispiele zu schließen.

Beobachtungen:

- Entscheidungsbaumsuche findet im Raum *aller* Hypothesen statt.
  - ➜ Zielkonzept muss im Hypothesenraum sein
- Der ID3-Algorithmus trifft nur soviel Entscheidungen wie es Merkmale gibt.
  - ➜ kein Backtracking
  - ➜ *lokale* Optimierung von Entscheidungsbäumen

Worin sich der induktive Bias des ID3-Algorithmus manifestiert:

- Bevorzugung kleinerer Entscheidungsbäume
- stärker diskriminierende Merkmale näher an der Wurzel

Ist das gerechtfertigt?

Bemerkungen:

- Sei $\mathbf{A}_i$ der (endliche) Wertebereich von Merkmal $A_i,\ i = 1, \ldots, p,$ und sei C der (endliche) Wertebereich des Konzeptes c. Dann kann der durch die Menge aller Entscheidungsbäume gebildete Hypothesenraum H aufgefasst werden als die Menge aller Funktionen $h,\ h : \mathbf{A}_1 \times \ldots \times \mathbf{A}_p \rightarrow C$.
- Der induktive Bias des ID3-Algorithmus macht ihn robust hinsichtlich verrauschter Daten.
- Der induktive Bias des ID3-Algorithmus ist von einer anderen Art als der induktive Bias des (Version-Space-) Candidate-Elimination-Algorithmus:
  1. Der von dem Candidate-Elimination-Algorithmus betrachtete Hypothesenraum H ist unvollständig; er stellt eine vergröberte (restriktive) Sicht des Raums aller Hypothesen dar, denn nur Konjunktionen von Attribut-Wert-Paaren sind als Hypothesen möglich. Dieser vergröberte Hypothesenraum H wird durch den Candidate-Elimination-Algorithmus vollständig durchsucht. Stichwort: *Restriction-Bias*
  2. Der von dem ID3-Algorithmus betrachtete Hypothesenraum H ist vollständig, weil alle – in Abhängigkeit der Merkmale und ihrer Ausprägungen – konstruierbaren diskretwertigen Funktionen (Hypothesen) durch einen Entscheidungsbaum repräsentierbar sind. Dieser vollständige Hypothesenraum H wird nur unvollständig (präferiert) durchsucht. Stichwort: *Preference-Bias* bzw. *Search-Bias*

Entscheidungsbaumalgorithmen

Klassifikationsproblem (Wiederholung)

- X sei ein Instanzenraum (Merkmalsraum) über endlich vielen Merkmalen mit beliebigem Skalennieveau.
- C sei eine Menge von Klassen.
- $c : X \rightarrow C$ sei der zu lernende Klassifikator für X.
- $D = \{(\mathbf{x}_1, c(\mathbf{x}_1)), \ldots, (\mathbf{x}_n, c(\mathbf{x}_n))\} \subseteq X \times C$ sei eine Menge von Lernbeispielen.

Aufgabe: Konstruktion eines Entscheidungsbaumes T für c auf Basis von D.

Entscheidungsbaumalgorithmen

Klassifikationsproblem (Wiederholung)

- X sei ein Instanzenraum (Merkmalsraum) über endlich vielen Merkmalen mit beliebigem Skalennieveau.
- C sei eine Menge von Klassen.
- $c : X \to C$ sei der zu lernende Klassifikator für X.
- $D = \{(\mathbf{x}_1, c(\mathbf{x}_1)), \ldots, (\mathbf{x}_n, c(\mathbf{x}_n))\} \subseteq X \times C$ sei eine Menge von Lernbeispielen.

Aufgabe: Konstruktion eines Entscheidungsbaumes T für c auf Basis von D.

CART-Algorithmus [Breiman 1984]:

1. Es sind nur binäre Splits bezüglich *eines* Merkmals erlaubt. Für Merkmal A mit Wertebereich $\mathbf{R}$ gilt:
$$X = \{\mathbf{x} \in X : \mathbf{x}|_A \leq 4\} \cup \{\mathbf{x} \in X : \mathbf{x}|_A > 4\}$$
2. Splits werden z.B. auf Basis des Gini-Index oder der Entropie berechnet.

Entscheidungsbaumalgorithmen

Binäre Splits

Jedes Element $\mathbf{x} \in X$ ist ein p-dimensionaler Merkmalsvektor $\mathbf{x} = (x_1, \ldots, x_p)$ mit Merkmalen ordinalen oder kategorischen Typs.

- Ein Split hängt nur vom Wert *eines* Merkmals aus $x_1, \ldots, x_p$ ab.
- Für ordinale Merkmale A_i sind Fragen folgender Art erlaubt:

  „Gilt $x_i \leq b$?“ mit $b \in]-\infty, \infty[$
- Für kategorische Merkmale A_i sind Fragen folgender Art erlaubt:

  „Gilt $x_i \in A_i'$?“ mit $A_i' \subset A_i$

Entscheidungsbaumalgorithmen

Binäre Splits

Jedes Element $\mathbf{x} \in X$ ist ein p-dimensionaler Merkmalsvektor $\mathbf{x} = (x_1, \ldots, x_p)$ mit Merkmalen ordinalen oder kategorischen Typs.

- Ein Split hängt nur vom Wert *eines* Merkmals aus $x_1, \ldots, x_p$ ab.
- Für ordinale Merkmale A_i sind Fragen folgender Art erlaubt:

  „Gilt $x_i \leq b$?“ mit $b \in]-\infty, \infty[$
- Für kategorische Merkmale A_i sind Fragen folgender Art erlaubt:

  „Gilt $x_i \in A_i'$?“ mit $A_i' \subset A_i$

Für eine Beispielsammlung $D = \{(\mathbf{x}_1, c(\mathbf{x}_1)), \ldots, (\mathbf{x}_n, c(\mathbf{x}_n))\}$ gibt es nur endlich viele unterschiedliche Splits verschiedener Güte:

- Für ordinale Merkmale A_i wähle b als Mitte zwischen zwei in D vorkommenden Werten für $x_{1,j}, \ldots, x_{n,j}$.
- Für kategorische Merkmale A_i wähle A_i' mit $0 < |A_i'| \leq |A_i \setminus A_i'|$.

Bemerkungen:

- Man kann Splits als Fragen „Ist $x \in A$?“ für ein $A \subset X$ auffassen. Durch die Frage werden die Elemente von $X(t)$ in $X(t_L)$ für Antwort „ja“ und $X(t_R)$ für Antwort „nein“ aufgeteilt.
- Für ein kategorisches Merkmal A_i benutzen wir A_i auch als Bezeichner des Grundbereichs der Werte für dieses Merkmal, z.B. $\mathit{Farbe} = \{\mathit{rot}, \mathit{gelb}, \mathit{gruen}\}$.

Entscheidungsbaumalgorithmen

Konstruktion eines Entscheidungsbaumes gemäß *TDIDT*

- Für jeden Knoten t des Entscheidungsbaumes wird eine Kandidatenmenge von möglichen binären Splits für t generiert.
- Man wählt ein Split-Kriterium mit maximaler Güte – also eines, das die Impurity am stärksten reduziert.

$$\Delta\iota = \iota(t) - p_L(\{D(t_L), D(t_R\}, t) \cdot \iota(t_L) - p_R(\{D(t_L), D(t_R\}, t) \cdot \iota(t_R)$$

mit $p_L(\{D(t_L), D(t_R\}, t) = \frac{p(t_L)}{p(t)}$ und $p_R(\{D(t_L), D(t_R\}, t) = \frac{p(t_R)}{p(t)}$.

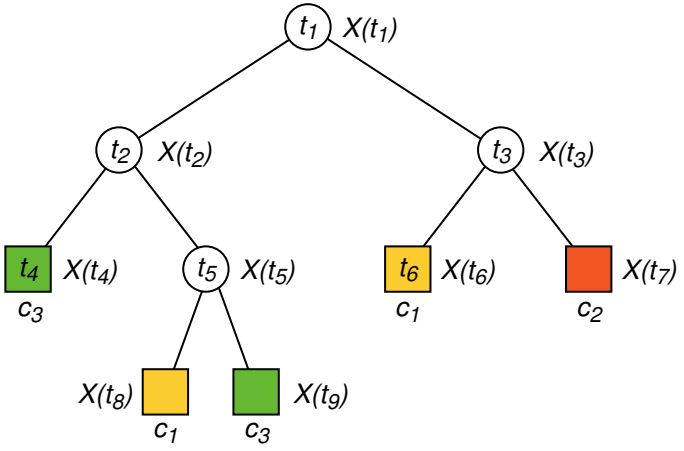
Beispiele für Impurity-Funktionen:

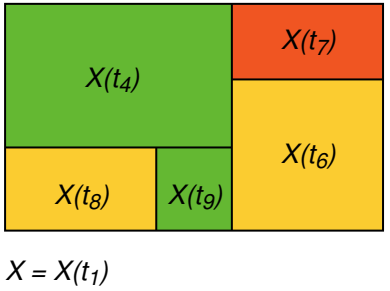
- Entropie: $i(t) = -\sum_{c \in C} p(c \mid X(t)) \log p(c \mid X(t))$
- Gini-Index: $i(t) = \sum_{c_1, c_2 \in C, c_1 \neq c_2} p(c_1 \mid X(t)) \cdot p(c_2 \mid X(t))$

Entscheidungsbaumalgorithmen

Konstruktion eines Entscheidungsbaumes gemäß *TDIDT*

Der Merkmalsraum X bestehe aus den Punkten der Ebene; das heißt es gibt zwei reellwertige Merkmale. Illustration:





Durch sukzessives Splitting wird der Merkmalsraum X in achsenparallele Rechtecke aufgeteilt.

Bemerkungen:

- Eine Erweiterung der CART-Entscheidungsbäume sind die sogenannten schiefen Entscheidungsbäume *(Oblique Decision Trees)*. Während in CART-Knoten nur Bedingungen der Art $x_i \geq b$ möglich sind, erlauben schiefe Entscheidungsbäume auch Linearkombinationen von Merkmalsausprägungen $\lambda_1 x_1 + \ldots + \lambda_p x_p \geq b$.

Kapitel ML: III (Fortsetzung)

III. Entscheidungsbäume

- Repräsentation und Konstruktion
- Splitting
- Entscheidungsbaumalgorithmen
- Pruning

Pruning

Missklassifikationskosten beim Splitting

Lemma 3

Für jeden Split $D(t_1), \ldots, D(t_s)$ einer Menge von Lernbeispielen $D(t)$ in einem Entscheidungsbaum T gilt:

$$\mathit{Err}(D(t)) \geq \sum_{i \in \{1,\ldots,s\}} \mathit{Err}(D(t_i)),$$

mit Gleichheit in dem Fall, dass allen Knoten $t, t_1, \ldots, t_s$ dieselbe Klasse zugewiesen wird.

Die Folge ist eine Bevorzugung großer Bäume, d.h. eine weitestmögliche Zerlegung der Menge D von Lernbeispielen.

Pruning

Missklassifikationskosten beim Splitting (Fortsetzung)

Beweisidee:

$$
\begin{aligned}
\mathit{Err}(D(t)) &= \min_{c' \in C} \sum_{c \in C} \mathit{cost}(c' \mid c) \cdot p(c \mid t) \cdot p(t) \\
&= \sum_{c \in C} \mathit{cost}(\mathit{class}(t) \mid c) \cdot p(c, t) \\
&= \sum_{c \in C} \mathit{cost}(\mathit{class}(t) \mid c) \cdot (p(c, t_1) + \ldots + p(c, t_{k_s})) \\
&= \sum_{i \in \{1,\ldots,k_s\}} \sum_{c \in C} \mathit{cost}(\mathit{class}(t) \mid c) \cdot (p(c, t_i)
\end{aligned}
$$

$$
\mathit{Err}(D(t)) - \textstyle\sum_{i \in \{1,\ldots,k_s\}} \mathit{Err}(D(t_i)) =
$$

$$
\sum_{i \in \{1,\ldots,k_s\}} \left(\sum_{c \in C} \mathit{cost}(\mathit{class}(t) \mid c) \cdot (p(c, t_i) - \min_{c' \in C} \sum_{c \in C} \mathit{cost}(c' \mid c) \cdot p(c, t_i) \right)
$$

Jeder Summand der rechten Seite ist größer als Null.

Pruning

Overfitting

Definition 18 (Überanpassung *(Overfitting)*)
Sei X die Menge aller interessierenden Beispiele, D eine Menge von Lernbeispielen und H ein Hypothesenraum. Dann stellt $h \in H$ eine Überanpassung *(Overfitting)* von D dar, falls ein $h' \in H$ mit folgender Eigenschaft existiert:

$$\mathit{Err}(D,h) < \mathit{Err}(D,h') \quad \wedge \quad \mathit{Err}(X,h) > \mathit{Err}(X,h')$$

Hierbei stellen $\mathit{Err}(D,h)$ und $\mathit{Err}(X,h)$ monoton steigende Funktionen in der Anzahl der mit einer Hypothese inkonsistenen Beispiele dar.

Pruning

Overfitting

Definition 18 (Überanpassung *(Overfitting)*)
Sei X die Menge aller interessierenden Beispiele, D eine Menge von Lernbeispielen und H ein Hypothesenraum. Dann stellt $h \in H$ eine Überanpassung *(Overfitting)* von D dar, falls ein $h' \in H$ mit folgender Eigenschaft existiert:

$$\mathit{Err}(D,h) < \mathit{Err}(D,h') \quad \wedge \quad \mathit{Err}(X,h) > \mathit{Err}(X,h')$$

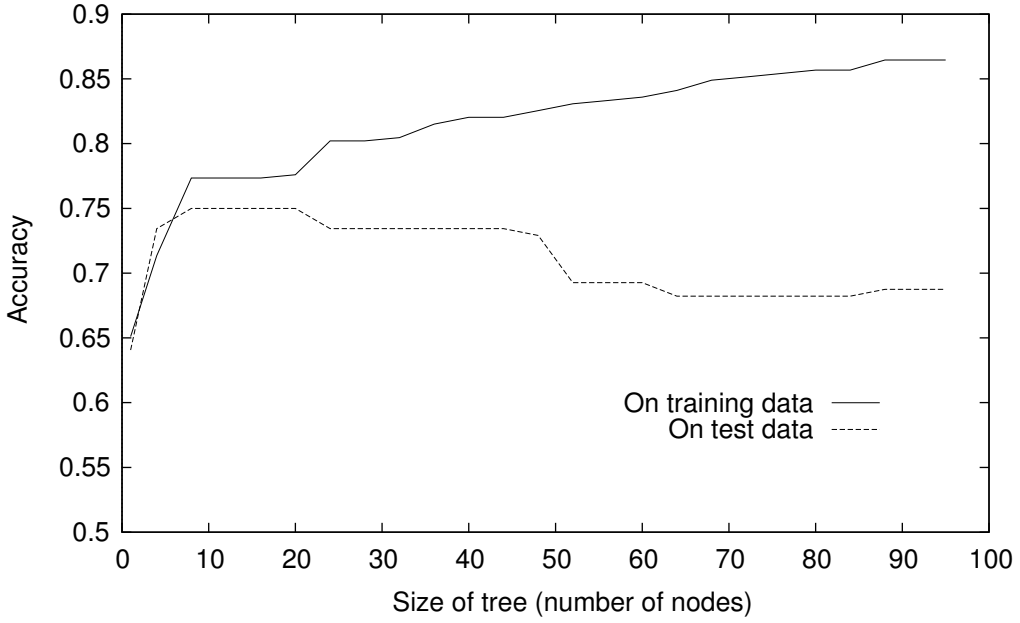
Hierbei stellen $\mathit{Err}(D,h)$ und $\mathit{Err}(X,h)$ monoton steigende Funktionen in der Anzahl der mit einer Hypothese inkonsistenen Beispiele dar.

Ursachen für Overfitting:

- die Trainingsmenge ist verrauscht,
- die Trainingsmenge ist nicht repräsentativ,
- zu kleine Trainingsmenge täuscht nicht vorhandene Regelmäßigkeiten vor,
- ...

Pruning

Overfitting



[Mitchell 1997]

Bemerkungen:

- Der Trainingsfehler $Err(D, T)$ für den Entscheidungsbaum T fällt monoton.
- Wann wird der Wert 0 erreicht?
- Wo ist $Err(D, T)$ am genauesten in Bezug auf $Err^*(T)$?
- Die Missklassifikationsrate $Err(D_V, T)$ auf einer Testmenge D_V hat ein lokales Minimum, häufig bei relativ geringer Blattzahl.

Pruning

Overfitting

Ansätze, um Overfitting entgegenzuwirken:

1. Beschränkung der Entscheidungsbaumkonstruktion während des Trainings
2. Verkleinerung (Pruning) von Entscheidungsbäumen nach dem Training
   - Aufteilung der Menge D in Trainings-, Validierungs- und Testmenge
     (a) Reduced-Error-Pruning
     (b) Minimal Cost-Complexity Pruning
     (c) Rule-Post-Pruning
   - statistische Tests (z. B. χ^2), um Generalisierbarkeit zu beurteilen
   - heuristische Minimierung der Entscheidungsbaums

Pruning

Stopping

Einfache Kriterien:

- Größe von $D(t)$
  Ein Knoten t wird nicht weiter zerlegt, wenn die Anzahl der Beispiele in $D(t)$ zu gering ist: $|D(t)| < \beta$.
- Reinheit von $D(t)$
  Ein Knoten t wird nicht weiter zerlegt, wenn Impurity nicht stark genug reduziert werden kann: $\Delta\iota(\{D(t_1), \ldots, D(t_s)\}, t) < \beta$ für alle Splits.

Problem:

- Abbruch des Splitting mit kleinem β führt zu übergroßen Entscheidungsbäumen.
- Abbruch des Splitting mit großem β kann nützliche Splits verhindern.

Pruning

Pruning statt Stopping

Idee:

- Bilde einen ausreichend großen Entscheidungsbaum T_{max} und
- beschneide T_{max} „auf die richtige Weise“ von Blättern in Richtung Wurzel.

Wann ist T_{max} groß genug?

Aufbau von T_{max} ist beendet, wenn jeder Blattknoten

- eine genügend kleine Teilmenge der Lernmenge D repräsentiert oder
- nur Beispiele einer Klasse repräsentiert (Knoten ist pur) oder
- nur Beispiele mit identischen Merkmalsvektoren repräsentiert.

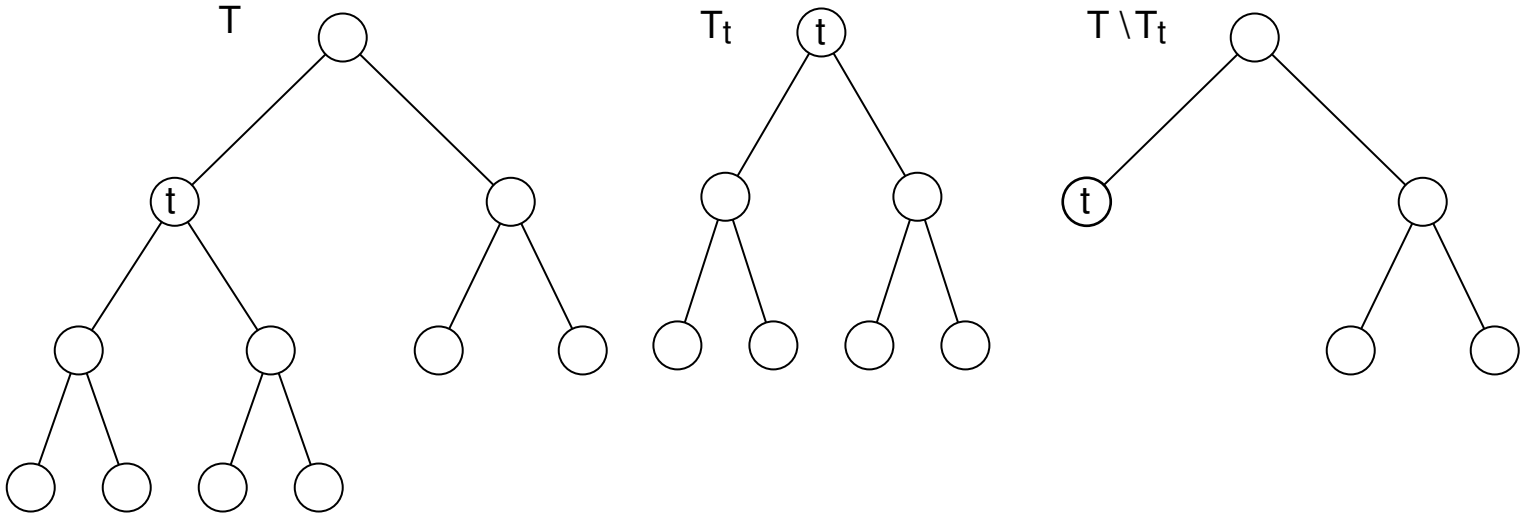
Bemerkungen:

- Optimal für T_{max}: „genügend klein“ $\Leftrightarrow N(t) = 1$
- Kompromiss für T_{max}: „genügend klein“ $\Leftrightarrow N(t) \leq N_{min}$, z.B. $N_{min} = 5$
- Der Trainingsfehler $\mathit{Err}(D, T)$ ist als Schätzer für $\mathit{Err}^*(T)$ ungeeignet wegen seiner Monotonie-Eigenschaft. Daher muss die „richtige“ Größe des Entscheidungsbaumes auf andere Weise bestimmt werden.

Pruning

Terminologie für Teilbäume

Baum T mit Knoten t, Zweig T_t, Resultat des Pruning von T_t



„t“ bezeichnet den Teilbaum mit Wurzel t.

Pruning

Definition 19 (Pruning)
Für einen Baum T mit innerem Knoten t besteht das Pruning des Zweiges T_t aus der Entfernung aller Nachfolgerknoten von t und der durch diese Elimination betroffenen Kanten. Der so beschnittene Baum wird mit $T \setminus T_t$ bezeichnet. Der Knoten t ist in $T \setminus T_t$ ein Blattknoten.

Definition 20 (Pruning-induzierte Ordnung)
Entsteht ein Baum T' durch (mehrfaches) Pruning aus einem Baum T, schreiben wir kurz $T' \preceq T$. Die Relation $\preceq$ bildet eine Halbordnung auf der Menge der Bäume.

Pruning

Definition 19 (Pruning)
Für einen Baum T mit innerem Knoten t besteht das Pruning des Zweiges T_t aus der Entfernung aller Nachfolgerknoten von t und der durch diese Elimination betroffenen Kanten. Der so beschnittene Baum wird mit $T \setminus T_t$ bezeichnet. Der Knoten t ist in $T \setminus T_t$ ein Blattknoten.

Definition 20 (Pruning-induzierte Ordnung)
Entsteht ein Baum T' durch (mehrfaches) Pruning aus einem Baum T, schreiben wir kurz $T' \preceq T$. Die Relation $\preceq$ bildet eine Halbordnung auf der Menge der Bäume.

Problematik:

- Kandidaten-Teilbäume sind nicht angeordnet bzgl. „$\preceq$“.
- Kandidaten-Teilbäume entstehen meist nicht durch wiederholtes Pruning von T_{max}.

Welches Maß kann die Güte der durch Pruning entstehenden Kandidaten bewerten?

Pruning

Pruning mit Validierungsmenge

Reduced-Error-Pruning:

1. Pruning des Zweiges T_t für einen inneren Knoten t.
2. *label*(t) ist das häufigste Konzept der mit T_t assoziierten Beispiele, d.h. der Beispiele in $D(t)$.
3. Falls *Err*$(D_V, T) <$ *Err*$(D_V, T \setminus T_t)$, Rücknahme der Ersetzung.
   (D_V ist fest gewählte Validierungsmenge.)
4. Weiter bei (1), bis alle Elternknoten der Blattknoten des aktuellen Entscheidungsbaums überprüft wurden.

Pruning

Pruning mit Validierungsmenge

Reduced-Error-Pruning:

1. Pruning des Zweiges T_t für einen inneren Knoten t.
2. *label*(t) ist das häufigste Konzept der mit T_t assoziierten Beispiele, d.h. der Beispiele in $D(t)$.
3. Falls *Err*$(D_V, T) <$ *Err*$(D_V, T \setminus T_t)$, Rücknahme der Ersetzung.
   (D_V ist fest gewählte Validierungsmenge.)
4. Weiter bei (1), bis alle Elternknoten der Blattknoten des aktuellen Entscheidungsbaums überprüft wurden.

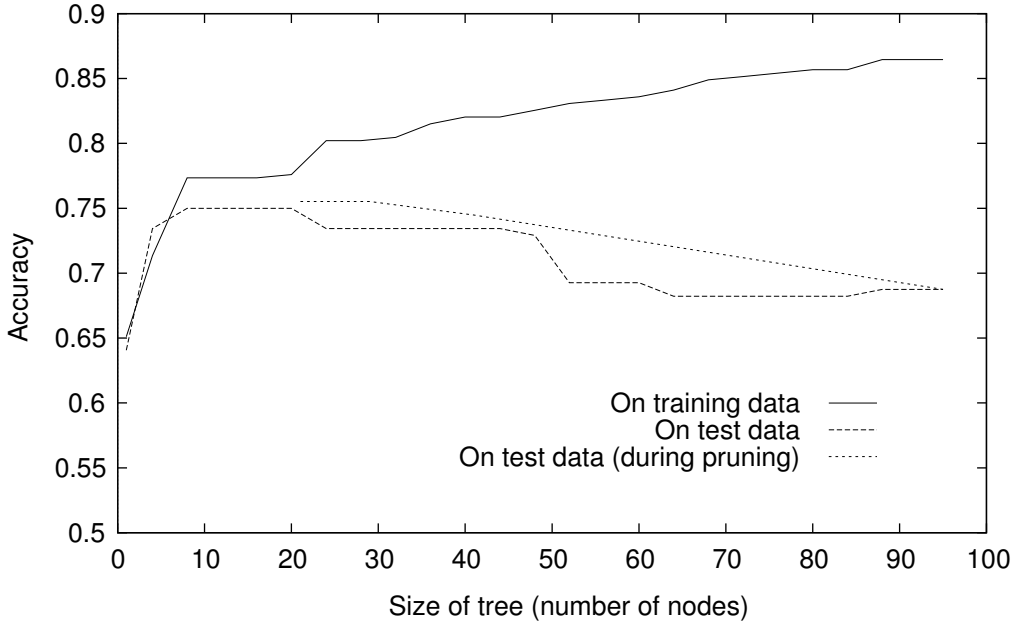
Nachteil:

Ist D klein, so geht durch eine Aufteilung in Trainings- und Validierungsmenge wertvolle Information zur Konstruktion des Entscheidungsbaums verloren.

Verbesserung: Rule-Post-Pruning

Pruning

Pruning mit Validierungsmenge



[Mitchell 1997]

Pruning

Pruning mit Maß für Kosten-Komplexität

Einfache Idee:

- Teile die von T_{max} aus durch Pruning erreichbaren Teilbäume auf Teilmengen auf nach der Anzahl ihrer Blätter.
- Wähle aus den Teilmengen jeweils einen Teilbaum mit minimalem Wert für $\mathit{Err}(D, T)$ als Kandidaten.

Pruning

Pruning mit Maß für Kosten-Komplexität

Einfache Idee:

- Teile die von T_{max} aus durch Pruning erreichbaren Teilbäume auf Teilmengen auf nach der Anzahl ihrer Blätter.
- Wähle aus den Teilmengen jeweils einen Teilbaum mit minimalem Wert für $\mathit{Err}(D, T)$ als Kandidaten.

Definition 21 (Kostenkomplexität)
Für einen Baum T wählen wir mit $|\mathit{leaves}(T)|$, also der Anzahl der Blattknoten in T, als Maß für die Komplexität (complexity) von T.

Für einen Komplexitätsparameter $\alpha \in \mathbf{R}, \alpha \geq 0$ bezeichnet das Maß

$$\mathit{Err}_\alpha(D, T) := \mathit{Err}(D, T) + \alpha|\mathit{leaves}(T)|$$

die Kostenkomplexität von T bei einer Beispielmenge D.

Pruning

Definition 22 (kleinster minimierender Teilbaum)

Ausgehend von einem Baum T_{max} und einem Komplexitätsparameter $\alpha \in \mathbf{R}, \alpha \geq 0$ bezeichnen wir einen Baum $T(\alpha) \preceq T_{max}$ als kleinsten minimierenden Teilbaum, wenn er folgende Bedingungen erfüllt:

1. $\mathit{Err}_\alpha(D, T(\alpha)) = \min_{T \preceq T_{max}} \mathit{Err}_\alpha(D, T)$
2. Falls für einen Teilbaum $T \preceq T_{max}$ gilt $\mathit{Err}_\alpha(D, T(\alpha)) = \mathit{Err}_\alpha(D, T)$, so folgt $T(\alpha) \preceq T$.

Pruning

Definition 22 (kleinster minimierender Teilbaum)

Ausgehend von einem Baum T_{max} und einem Komplexitätsparameter $\alpha \in \mathbf{R}, \alpha \geq 0$ bezeichnen wir einen Baum $T(\alpha) \preceq T_{max}$ als kleinsten minimierenden Teilbaum, wenn er folgende Bedingungen erfüllt:

1. $\mathit{Err}_\alpha(D, T(\alpha)) = \min_{T \preceq T_{max}} \mathit{Err}_\alpha(D, T)$
2. Falls für einen Teilbaum $T \preceq T_{max}$ gilt $\mathit{Err}_\alpha(D, T(\alpha)) = \mathit{Err}_\alpha(D, T)$, so folgt $T(\alpha) \preceq T$.

Lemma 4

Für jeden Baum T und jeden Komplexitätsparameter $\alpha \in \mathbf{R}, \alpha \geq 0$ existiert ein kleinster minimierender Teilbaum $T(\alpha) \preceq T$.

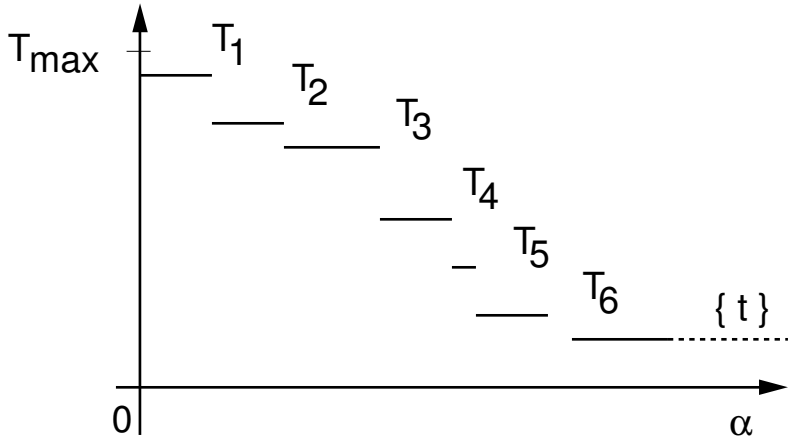
Beweisidee

Induktion für $\mathit{Err}_\alpha(D, T(\alpha)) = \min\{\mathit{Err}_\alpha(D, t), \mathit{Err}_\alpha(D, T_L) + \mathit{Err}_\alpha(D, T_R)\}$ mit t Wurzel von T sowie T_L, T_R als linker und rechter Teilbaum von t.

Pruning

Pruning mit Maß für Kosten-Komplexität

Beobachtung:



Müssen alle Werte für α durchlaufen werden? Sind die Teilbäume geordnet?

Pruning

Weakest Link Pruning

1. Bestimmung von $T_1 = T(\alpha_1)$ mit $\alpha_1 = 0$ ausgehend von T_{max}

   Wegen $R(T_1) = R(T_{max})$

   und $R(t) \geq R(t_L) + R(t_R)$
   für jeden inneren Knoten t von T_{max} mit Nachfolgern t_L und t_R

   schneiden wir alle Blätter t'_L und t'_R ab, für die mit ihrem Vorgänger t' gilt $R(t') = R(t'_L) + R(t'_R)$.

Pruning

Weakest Link Pruning (Fortsetzung)

2. Bestimmung von T_{k+1} ausgehend von T_k

   Für jeden inneren Knoten t von T_k gilt $R(t) > R(T_t)$. Setzt man

   $$R_\alpha(\{t\}) := R(t) + \alpha \qquad \text{und} \qquad R_\alpha(T_t) := R(T_t) + \alpha \cdot |\widetilde{T}_t|$$

   so gilt $R_\alpha(T_t) < R_\alpha(\{t\})$ für genügend kleine Werte für α.

   Setze

   $$g_k(t) := \begin{cases} \frac{R(t)-R(T_t)}{|\widetilde{T}_t|-1} & \text{für } t \text{ innerer Knoten von } T_k \\ +\infty & \text{für } t \text{ Blattknoten von } T_k \end{cases}$$

   und $\alpha_{k+1} := \min_{t \in T_k} g_k(t)$.

   Die Knoten $\bar{t}_k \in \operatorname{argmin}_{t \in T_k} g_k(t)$ bezeichnet man als schwächste Knoten (weakest links) in T_k. Durch Pruning aller dieser Zweige $T_{\bar{t}_k}$ (weakest link pruning) erhalten wir T_{k+1}.

Pruning

Weakest Link Pruning (Fortsetzung)

3. Bewertung der Teilbäume T_k

   Bestimme *Err*(D_V, T_k) für alle Teilbäume T_k und wähle den Teilbaum mit minimalem Fehler auf der Validierungsmenge D_V.

Pruning

Weakest Link Pruning (Fortsetzung)

3. Bewertung der Teilbäume T_k

   Bestimme *Err*(D_V, T_k) für alle Teilbäume T_k und wähle den Teilbaum mit minimalem Fehler auf der Validierungsmenge D_V.

Satz 6

Die durch das Weakest Link Pruning erhaltene endliche Folge (α_k) ist streng monoton wachsend, d.h.

$$\alpha_k < \alpha_{k+1} \qquad \text{für alle } k$$

und es gilt

$$T(\alpha) = T(\alpha_k) = T_k \qquad \text{für alle } k \text{ und } \alpha_k \leq \alpha < \alpha_{k+1}$$

sowie

$$T_1 \succeq T_2 \succeq T_3 \succeq T_4 \ldots \succeq \{t\} \qquad \text{mit } t \text{ Wurzel von } T_{max}.$$

Pruning

Erweiterungen

- Rule-Post-Pruning
- Berücksichtigung von Missklassifikationskosten beim Split
- Surrogate Splits für fehlende Merkmalwerte
- Linearkombinationen von Merkmalen
- Regression Trees