

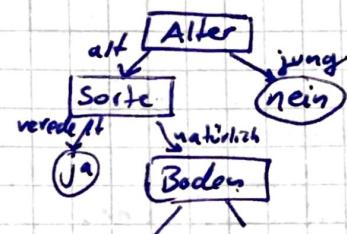
# Maschinelles Lernen I Entscheidungsbäume (Topic #)

(1)

Entscheidungsbäume dienen der Darstellung von Entscheidungsregeln und sind geordnete und gerichtete Bäume. Sie sind eine Methode zur autom. Klassifikation von Daten und jeder Knoten repräsentiert eine logische Regel und jedes Blatt eine Antwort auf das Entscheidungsproblem.

## Repräsentation

- jeder Knoten = ein Attributtest
- jeder Zweig = ein Attributwert
- jedes Blatt = eine Aussage (Klassifikation)



## Beschreibung allgemein:

Disjunktion (oder) von Konjunktionen (und) von Bedingungen an die Attributwerte einer Instanz, z.B.

(Vorhersage = sonnig  $\wedge$  Luftfeuchtigkeit = normal)  $\vee$  (Vorhersage = bedeckt)  
 $\vee$  (Vorhersage = regnerisch  $\wedge$  Wind = schwach)

Wann sind Entscheidungsbäume geeignet? Wenn...

- Instanzen sich als Attribut-Wert Paare schreiben lassen
- Zielfunktion diskrete Ausgabewerte besitzt
- disjunkte Hypothesen erforderlich
- Beispieldaten verrauscht sind
- Beispieldaten fehlende Attributwerte enthalten

## Lernen von EB: ID3 (Top Down Aufbau)

- 1) A → bestes Attribut \*
- 2) weise A als Attributtest nächsten Knoten zu
- 3) füge für jeden möglichen Wert von A einen Nachfolgeknoten ein
- 4) verteile Train Examples gemäß Werte auf Nachfolgeknoten
- 5) wenn Examples perfekt klassifiziert  $\rightarrow$  stop  
sonst iteriere über Nachfolgeknoten

## Entropie als Maß des Informationsgehalts in den Trainingsdaten:

$$\text{Entropie}(S) = -P_0 \log_2 P_0 - P_1 \log_2 P_1$$

mit  $S$ : Menge d. Train Examples

$P_0$ : Anteil d. positiven Beispiele in  $S$

Bestes Attribut wird mithilfe Gewinn bestimmt:

$$* \text{Gewinn}(S, A) = \text{Entropie}(S) - \sum_{v \in V(A)} \frac{|S_v|}{|S|} \text{Entropie}(S_v)$$

$P_0$ : Anteil d. negativen Beispiele in  $S$   
mit  $V(A)$ : Menge aller mögl. Attributwerte von A

$S_v$ : Untermenge von  $S$ , für die A den Wert  $v$  annimmt

(Ziel ist es, die Daten durch Festhalten eines Attributwertes v möglichst die Klasse 0 oder 1 einzuteilen, d.h. sukzessive die Entropie maximal zu reduzieren)

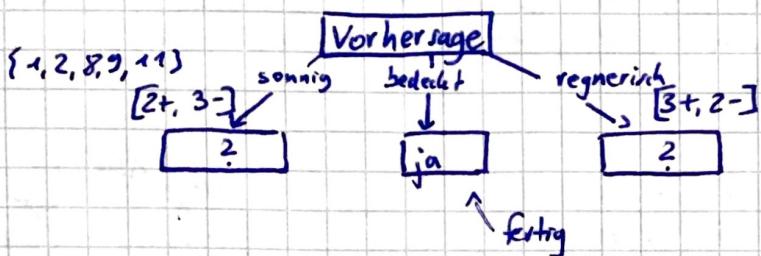
Gewinn( $S, A$ ) → erwartete Reduzierung der Entropie durch Einsortierung über Attribut A

Beispiel →

## siehe Foliensatz 7, S. 15) Beispiel:

(2)

Situation: „Vorhersage“ war letztes Attribut, Train Examples wurden gemäß Werte auf Nachfolgeknoten verteilt, Examples noch nicht perfekt klassifiziert → iteriere weiter



Möglichkeiten:

- Gewinn( $S_{\text{sonnig}}, \text{luftr. feucht.})$
- Gewinn( $S_{\text{sonnig}}, \text{Temp.}) = 0,57$
- Gewinn( $S_{\text{sonnig}}, \text{Wind}) = 0,19$

$$\text{Gewinn}(S_{\text{sonnig}}, \text{luftr. feucht.}) = \text{Entropie}(S_{\text{sonnig}}) - \sum_{v \in V} \frac{|S_{v(S)}}{|S_{\text{sonnig}}|} \cdot \text{Entropie}(S_v)$$

$$\text{Entropie}(S_{\text{sonnig}}) = -\frac{2}{5} \log_2 \frac{2}{5} - \frac{3}{5} \log_2 \frac{3}{5} = 0,97$$

$\text{sonnig} \wedge \text{Tennis} = \text{ja}$

$$V(\text{luftr. feucht.}) = \{\text{hoch}, \text{normal}\}$$

$$\text{für } v = \text{hoch}: \frac{|S_{\text{sonnighoch}}|}{|S_{\text{sonnig}}|} \cdot \text{Entropie}(S_{\text{sonnighoch}})$$

$$= \frac{3}{5} \cdot \left( -\frac{0}{3} \log_2 \frac{0}{3} - \frac{3}{3} \log_2 \frac{3}{3} \right) = 0$$

$\text{sonnig} \wedge \text{luftr. feucht.} = \text{ja}$

alle in einer Klasse

$$\text{für } v = \text{normal}: \frac{|S_{\text{sonnignormal}}|}{|S_{\text{sonnig}}|} \cdot \text{Entropie}(S_{\text{sonnignormal}})$$

$$= \frac{2}{5} \cdot 0 = 0$$

$$\text{Gewinn}(S_{\text{sonnig}}, \text{luftr. feucht.}) = 0,97 - 0 - 0 = 0,97$$

$\Rightarrow \text{Gewinn}(S_{\text{sonnig}}, \text{luftr. feucht.})$  die beste Wahl, da Gewinn am höchsten ( $\Leftrightarrow$  Entropie am niedrigsten)

## Suche im Hypothesenraum (ID3)

- Hypothesenraum ist bei Bäumen vollständig, d.h. Zielfunktion ist enthalten
- garantiert keine optimale Lösung, da es lokale Minima gibt (Hill-climbing-Algorithmus ist nicht konvex)

## Bias (Allgemein)

Was ist ein Restriktionsbias?

- > Menge der Hypothesen, für die man sich interessiert ( $H$ )
- > Einschränkung des Hypothesenraumes (z.B. nur EB)

Was ist ein Präferenzbias?

- > die präferierte Hypothese  $h$  der verfügbaren Hypothesen ( $h \in H$ )

Was ist ein induktiver Bias?

- > die Annahmen, die ein Lerner macht muss, um aus Trainingsbeispielen verallgemeinern zu können
- > d.h. gegeben Trainingabeispiele, bestimmte Annahmen müssen zum Gesehenen hinzugefügt werden, um die Outputs des Lerners in logische Schlussfolgerungen zu transformieren (Tom Mitchell)

Ein induktiver Bias kann sowohl Restriktions- als auch Präferenzbias beinhalten.

# 3 Maschinelles Lernen I Entscheidungsbäume (Topic #)

## Induktiver Bias bei ID3

Der induktive Bias bei ID3 impliziert einen Präferenzbias für bestimmte Hypothesen aber keinen Restriktionsbias, denn:

- der Hypothesenraum  $H$  wird nicht eingeschränkt  
( $H$  ist die Potenzmenge [alle Teilmengen der Grundmenge] der möglichen Instanzen [Attribut-Wert-Paare])
- ID3 präferiert:
  - kürzere Bäume
  - gute „Splits“ (Zer teilungen) nahe der Wurzel  
(Attribute mit hohem Gain weit oben)

## Occam's Razor - warum kurze Hypothesen bevorzugt werden sollten

Argumente:

- es gibt weniger kurze als lange Hypothesen!  
Eine kurze Hyp., welche die Daten korrekt beschreibt, ist wahrscheinlich kein Zufall. Eine lange Hyp., welche die Daten korrekt beschreibt, könnte hingegen Zufall sein.
- kurze Bäume sind effizienter bezüglich (interner) Repräsentation und Auswertung

## Ursachen und Probleme von Overfitting

(⇒ Overf. bei z.B.: Baum ist zu groß (zu komplexe Regeln), verletzt Occam's Razor)

### Formal Overfitting:

- Daten sind vervorscht, z.B.
  - eine Instanz gibt es zweimal mit unterschiedlichen Labels (<sonnig, warm, normal, stark> für Tennis = ja und Tennis = nein)  
⇒ Folge: Algorithmus kann ggf. nicht terminieren, weil Trainingsdaten nie perfekt klassifiziert sind (Abbruchkriterium)
  - Daten sind nicht repräsentativ
  - zu wenige Daten
    - keine Generalisierung, ineffizienter Baum

## Vermeidung von Overfitting

- 1) Frühzeitiges Stoppen des Baumwachstums (early stopping)
- 2) „Pruning“ (nachträgliches Schneiden des Baumes)

## Pruning

- Daten in Trainings- und Testdaten trennen
- 1) Trainiere Baum auf Trainingsdaten (als gäbe es kein Overfitting)
- 2) Evaluiere die Auswirkung auf Klassifikationsrate wenn ein Knoten entfernt wird (beginne bei Blättern)
  - a) wenn Rate sinkt, entferne Knoten, gehe weiter hoch im Baum
  - b) sonst beende Pruning

Problem: bei wenigen Daten führt Aufteilung in Train/Testdatensätze zu noch wenigen Daten → höhere Fehlerrate/Fehlerranfälligkeit des Baumes

Unterteile Daten in (variable) Trainings- und Testdaten. Trainiere  $n$  Bäume, teste auf Validierungssatz. Wähle Baum mit kleinstem Fehler (nach Georgia Tech-Kurs auch Methode des Prunings)

n  
(Cross Validation)

## Probleme bei ID3/Erweiterungen von ID3

1) Attribute mit vielen<sup>1</sup> Werten werden durch Gewinn(S,A) gegenüber solchen mit wenigen Werten bevorzugt.

Beispiel: Attribut „Kreditkartennummer“

→ identifiziert jeden Kunden eindeutig

=> hoher Gain (niedrige Entropie):  $-\frac{1}{200} \log_2 \frac{1}{200} - \frac{199}{200} \log_2 \frac{199}{200} \approx 0,045$

im Gegensatz:  $-\frac{40}{200} \log_2 \frac{40}{200} - \frac{160}{200} \log_2 \frac{160}{200} \approx 0,144$  (z.B. Kundengruppe)  
1-5 200 Datensätze

Lösung: Bias von Attributen mit vielen Werten reduzieren, indem Anzahl und Größe einer Verzweigung bei der Attributwahl berücksichtigt wird (Information Gain Ratio)

$$\text{Gewinn Anteil}(S,A) = \frac{\text{Gewinn}(S,A)}{\text{SplitInformation}(S,A)}$$

$$\text{SplitInformation}(S,A) = -\sum_{i=1}^c \frac{|S_i|}{|S|} \log_2 \frac{|S_i|}{|S|}$$

$$\text{Bsp.: } 200 \times \left( -\frac{1}{200} \log_2 \frac{1}{200} \right) \approx 0,045 \\ \text{vs.: } 5 \times \left( -\frac{40}{200} \log_2 \frac{40}{200} \right) \approx 0,144$$

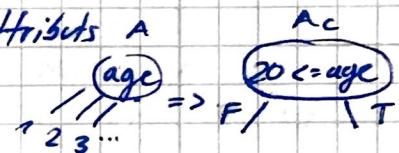
Gewinn fällt anteilig niedriger aus für viele Werte.

## 2) Kontinuierliche Werte

Gegeben: Attribut A mit kontinuierlichen Werten

Vorgehen: Automatische Definition eines neuen, diskret-wertigen Attributs A<sub>c</sub>

A<sub>c</sub>: A<sub>c</sub> ist wahr, wenn A > c



Aber: Wie wird c gewählt?

⇒ indem man mithilfe des Gewinns eine optimale Schwelle findet (opt. Schwelle liegt in der Mitte zw. zwei benachbarten Beispielen mit unterschiedlichen Klassen)

Beispiel: Temp. | 9° | 9° | 16° | 22° | 27° | 32°  
Tennis | nein | nein | ja | ja | ja | nein

Potentielle Schwellenwerte:

von den 2 Werten (9+16)  
ist keiner positiv aber alle negativ

$$1) (9 + 16)/2 = 12,5^\circ \rightarrow \text{Gewinn} = 1 - \frac{2}{6} * 0,00 - \frac{4}{6} * 0,81 = 0,34$$

$$2) (27 + 32)/2 = 29,5^\circ \rightarrow \text{Gewinn} = 1 - \frac{5}{6} * 0,97 - \frac{1}{6} * 0,00 = 0,08$$

(Höchster Informationsgewinn bei 1)  $\Rightarrow c = 12,5^\circ$

## 3) Unbekannte Attributwerte

Problem: wie unvollständige Daten (Attributwert fehlt) verwenden?

Lösung: fehlende AW bekommen:

- den häufigsten AW der Beispiele

- den häufigsten AW der Beispiele der gleichen Klasse

- jedem Wert v<sub>i</sub> mit Wahrsch. p<sub>i</sub> → Verteile das Beispiel gemäß p<sub>i</sub> anteilig auf Nachfolger → Schätzen der Verteilung der Daten

## 4) Attribute mit Kosten

Beispiel: medizinische Diagnose → Bluttest teurer als Pulsmessung

⇒ wie kann man einen mit den Trainingsdaten konsistenten Baum mit geringen Kosten lernen?

Ersetze Gewinn durch:  $\frac{\text{Gewinn}^2(S,A)}{\text{Kosten}(A)}$  oder  $\frac{2 \cdot \text{Gewinn}(S,A) - 1}{(\text{Kosten}(A) + 1)w}$

mit w ∈ [0,1]: gewichtung der Kosten

### ID3: Window-Verfahren

ist eine Lernmethode für große Datenmengen

Vorgehen:

- 1) wähle zufällig Teilmenge der Trainingsdaten aus (Window)
- 2) bestimme EB mit diesen Daten
- 3) verwende übrige Daten als Testdaten für gelernten EB
- 4) wenn nicht alle Daten korrekt klassifiziert → füge einen Teil (zufällig) der falsch klassif. zum Window hinzu → trainiere neu

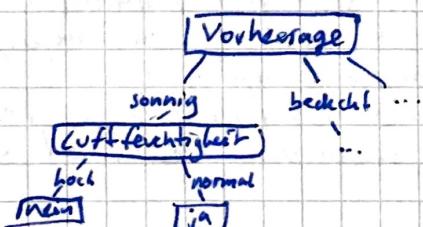
### C4.5

- Weiterentwicklung des ID3-Algorithmus (überwiegend ähnlich, siehe Erweiterungen ID3)
- unterstützt kontinuierliche Attributwerte
- kann mit fehlenden Attributwerten umgehen
- verwendet Rule Post-Pruning

Algorithmus:

- 1) Erstelle Baum aus Trainingsdaten wie bei ID3 mit Erweiterungen (verwende Information Gain Ratio, Overfitting ist erlaubt, fehlende AW mit Lücken einbeziehen, kont. AW mit Schwellen)
- 2) Konvertiere den Baum in Menge von IF-THEN-Regeln
  - IF-Teil: alle Attributtests einer Pfads
  - THEN-Teil: Ausgabe/Klasse
- 3) Wende „Pruning“ auf die Regeln an d.h. entferne Voraussetzungen der IF-Statements, solange Genauigkeit nicht sinkt
- 4) Sortiere alle Regeln nach Klassifikationsgüte und wandle sie in dieser Reihenfolge an

zu 3): Beispiel IF-THEN-Regeln



IF (Vorhersage = sonnig)  $\wedge$  (Luftf. = hoch)  
 THEN (Tennis = nein)  
 IF (Vorhersage = sonnig)  $\wedge$  (Luftf. = normal)  
 THEN (Tennis = ja)  
 usw.

zu 4): Abschätzung der Klassifikationsgüte

- Bestimmung Güte auf den Trainingsdaten
- Verwenden von statistischen Verfahren, z.B. Binomialverteilung ( $B(n, p, k)$ )  
 → führe Experiment (z.B. Urnenmodell) durch, da Schätzer (Kl.-rate) unbekannt  
 $n=1000$  (n mal ziehen),  $k=800$  (800 mal ist die Regel, z.B. „Vorhersage = sonnig dann Tennis = ja“ korrekt) mit Sicherheit z.B. 99,7% ( $>3\sigma$ )
- Standardabweichung berechnen
- Bestimme Konfidenzintervall z.B. mit Normalverteilung (Normalverteilung  $\mu \pm 3\sigma$  sein)

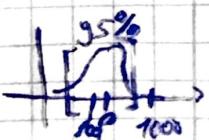
$$\text{rel. Häufigkeit } \frac{k}{n} = 0,8 = 80\%. \quad \frac{k}{n} \in [\mu - 3\sigma, \mu + 3\sigma]$$

$$|\frac{k}{n} - \mu| \leq 3\sigma \quad \dots \Rightarrow p_1: 0,76 \quad p_2: 0,82 \Rightarrow [0,76; 0,82] \text{ (Konf.-intervall)}$$

⇒ Verwendung der unteren Grenze des berechneten Konfidenzintervalls

99,7% aller Werte liegen zw.  $760$  und  $820 \Rightarrow$  Klassifikationsgenauigkeit zw. 76% und 82% bei dieser Regel für  $n=1000$  (zu 99,7%)

\*Binomialverteilung  
näherst Normalverteilung  
an für  $n \gg n$



## Vorteile durch das Umwandeln des Baums in Regeln:

(6)

- für einen Entscheidungsknoten können verschiedene Konflikte unterschieden werden (eine Regel für jeden Pfad im Baum)
- einfacheres Prüfen, da nicht unterschieden wird zwischen Attributen nah an der Wurzel und nah an den Blättern
- lesbar für Menschen

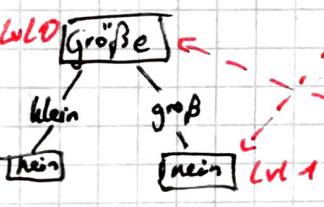
## 1) 5R

- inkrementelles Verfahren (ID3 u. C4.5 nicht inkrementell), d.h. sukzessives Einbringen von Beispielen

- Ergebnis äquivalent zu einem von ID3 erzeugter Baum

(Idee: es gibt Blätter (Antwortknoten) und Knoten (Entscheidungsknoten))

- Antwortknoten beinhalten a) Klassenbezeichner (ja/nein z.B.) und b) die Menge der Instanzbeschreibungen die zu dieser Klasse gehören
- Entscheidungsknoten besitzen einen Attributtest für jeden Attributwert und für jeden Attributwert positive und negative Counts für die Beispiele
- Zusätzlich: die Menge der noch nicht durchgeführten Attributtests an dem Knoten mit jeweils positiv/negativ-Counts für alle Attributwerte des Attributs



Algorithmus: (EB: bestehender EB, I: neu eingefügte Beispiele) Situation: I wird eingefügt in EB

- 1) Wenn EB leer, füge Blatt/Antwortknoten mit der Klasse von I ein
- 2) Wenn Antwortknoten der selben Klasse in EB existiert, füge I zur Menge der Instanzen dieses Blatts hinzu

3) Sonst (z.B. I ist von anderer Klasse) (AK)

- a) Wenn der EB aus nur dem Antwortknoten besteht ("unexpanded form"), dann wandle ihn um in einen Entscheidungsknoten mit beliebigem Testattribut
- b) aktualisiere Zähler des EK (für Testattribut und allen anderen, ungeprägten Attributen)
- c) Ist Testattribut nicht optimal (nicht niedrigste Entropie), dann
  - i. restrukturiere EB, dass Attribut mit niedrigster Entropie Testattribut wird
  - ii. restrukturiere Unterbäume des EB, sodass bestes Testattribut oben (tue dies nicht für 3)d) - Unterbaum)
- d) Aktualisiere rekursiv den gemäß der AW von Instanz I gewählten Unterbaum. Erweitere ggf. die Zweige.

# ② Maschinelles Lernen I Entscheidungsbäume (Topic 7)

## Beispiel zu ID3R-Algorithmus

Syntax: < Klasse, Größe, Haarfarbe, Augenfarbe >

- ① Daten: < -, klein, blond, braun > 1.  
< -, groß, dunkel, braun > 2.

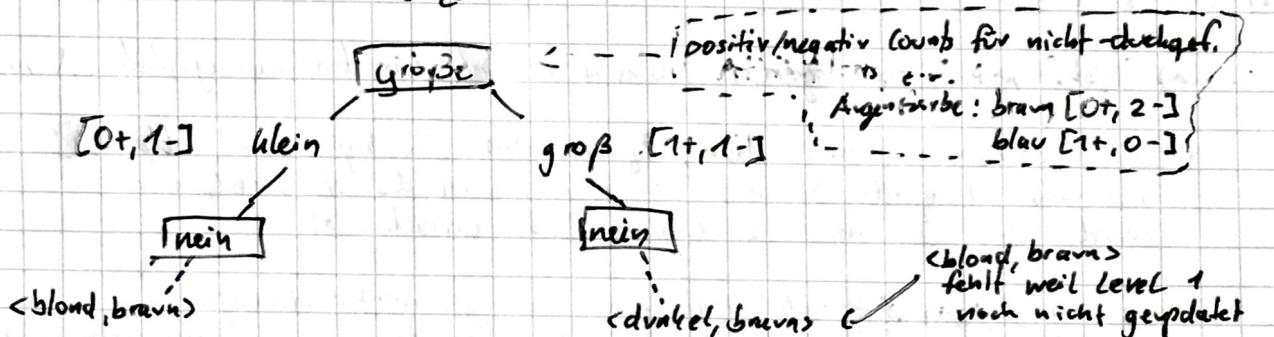
Schritt 1+2 ausführen: AK einfügen (wegen 1.), Instanz einfügen (wg. 2.)

[nein]

< klein, blond, braun >, < groß, dunkel, braun >

- ② Daten: < +, groß, blond, blau >

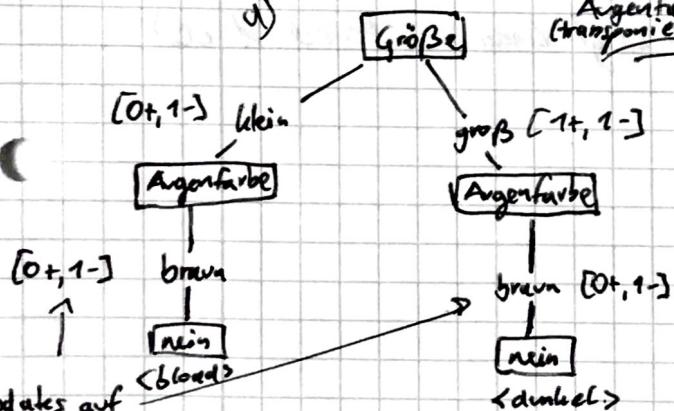
Schritt 3a,b): AK umwandeln in EK mit beliebigem Testattribut (hier: Größe), aktualisiere zähler



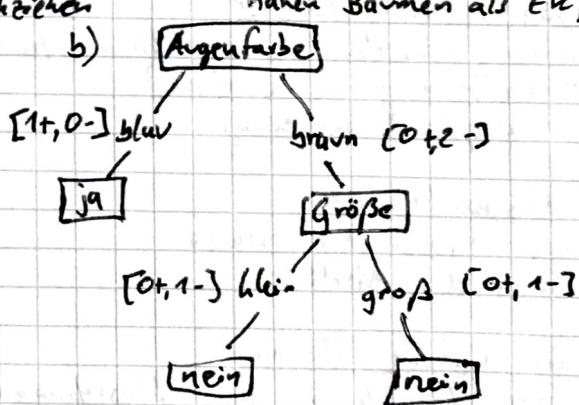
- ③ Daten: keine neuen Daten

Schritt 3c: Attribut Augenfarbe hat niedrigste Entropie (Information entnommen aus der Menge ungetesteter Attributwerk, günstigst im EK „Größe“) → Restrukturierung (Setze dieses Attribut in den nahen Bäumen als EK, siehe a))

a)



b)



Updates auf  
Level 1 nach  
nicht durchgefertigt,  
Level 0 hat Update  
bereits Schammonen

④ Schritt 3c,d: Unterbäume rekursiv aktualisieren

## Random Forests

(8)

- mehrere Entscheidungsbäume (Ensemble) bilden einen Wald
- für eine Klassifikation darf jeder EB in diesem Wald eine Entscheidung treffen, die Klasse mit den meisten Stimmen entscheidet endgültiges Ergebnis
- der Algorithmus hat zwei „Randomness“ - Quellen
  - 1) Bootstrap Samples: die Daten für einen Baum werden zufällig gezogen
  - 2) Attribute: für jeden Splitpunkt im EB wird eine neue Untermenge der Attribute zufällig gewählt  
=> Was passiert wenn alle EB des Forests alle Attribute betrachten und dann entscheiden könnten? Bei einem oder mehreren sehr „aussagekräftigen“ Attributen würden viele der EB des Walds diese wählen und miteinander korrelieren.

### Algorithmus:

Anzahl EB

#### 1) Für $b=1$ bis $B$

- a) Ziehe, mit Zurücklegen,  $n$  Trainingsbeispiele aus den Trainingssätzen  $X$  mit  $X = x_1, \dots, x_n$
- b) Lerne einen Random-Forest Baum  $T_b$  mit den Bootstrap-Daten, indem rekursiv folgende Schritte für jeden Knoten des EB wiederholt werden bis die minimale Knotenzahl erreicht ist
  - i) Wähle  $m$  Attribute zufällig von der Gesamtattributmenge
  - ii) Wähle das beste Attribut/Splitpunkt unter den  $m$  aus (Information Gain/Gewinn)
  - iii) Teile den Knoten in zwei Töchterknoten

#### 2) Ausgabe des EB-Ensembles $\{T_b\}_{b=1}^B$

### Eigenschaften von Random Forests

- schnelles Training (einzelne EB sind kleiner, Trainingzeit steigt linear mit Anzahl d. EB)
- parallelisierbar (Training + Evaluation)
- Effizient für große Datensätze

### Vergleich zu Standardbäumen:

- kein Pruning  $\rightarrow$  Overfitting erlaubt
- Randomisierung der Attribute

### Anforderungen an erstellte Bäume:

- jeder EB sollte für sich ein guter Klassifikator sein
- EB sollten unkorreliert sein untereinander