Universität Potsdam
Institut für Informatik
Lehrstuhl Maschinelles Lernen

Entscheidungsbäume

Christoph Sawade /Niels Landwehr/Paul Prasse
Silvia Makowski
Tobias Scheffer

Entscheidungsbäume

- Eine von vielen Anwendungen: Kreditrisiken

Entscheidungsbäume

- Einfach zu interpretieren.
- Liefern Klassifikation plus Begründung.
  - „Abgelehnt, weil weniger als 3 Monate beschäftigt und Kredit-Sicherheiten < 2 x verfügbares Einkommen“.
- Können aus Beispielen gelernt werden.
  - Einfacher Lernalgorithmus.
  - Effizient, skalierbar.
- Klassifikations- und Regressionsbäume.
- Klassifikations-, Regressions-, Modellbäume häufig Komponenten komplexer (z.B. Risiko-)Modelle.

Klassifikation

- Eingabe: Instanz (Objekt) $\mathbf{x} \in X$.
  - Instanzen sind durch Vektoren von Attributen repräsentiert.
  - Eine Instanz ist eine Belegung der Attribute.
  - Instanzen werden auch als Merkmalsvektoren bezeichnet.
  - $\mathbf{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_m \end{pmatrix}$
- Ausgabe: Klasse $y \in Y$; endliche Menge Y.
  - z.B {akzeptiert, abgelehnt}; {Spam, Nicht-Spam}.
  - Klasse wird auch als Zielattribut bezeichnet.

Klassifikationslernen

- Eingabe: Trainingsdaten.
  - $L = \langle (\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N) \rangle$
  - $\mathbf{x}_i = \begin{pmatrix} x_{i1} \\ \vdots \\ x_{im} \end{pmatrix}$
- Ausgabe: Klassifikator.
  - $f : X \rightarrow Y$

Z.B. Entscheidungsbaum:
Pfad entlang der Kanten zu
einem Blatt liefert Klassifikation

Regression

- Eingabe: Instanz (Objekt) $\mathbf{x} \in X$.
  - Instanzen sind durch Vektoren (fett gedruckt) von Attributen (kursiv) repräsentiert.
  - Eine Instanz ist eine Belegung der Attribute.
- Ausgabe: Funktionswert $y \in Y$; kontinuierlicher Wert.

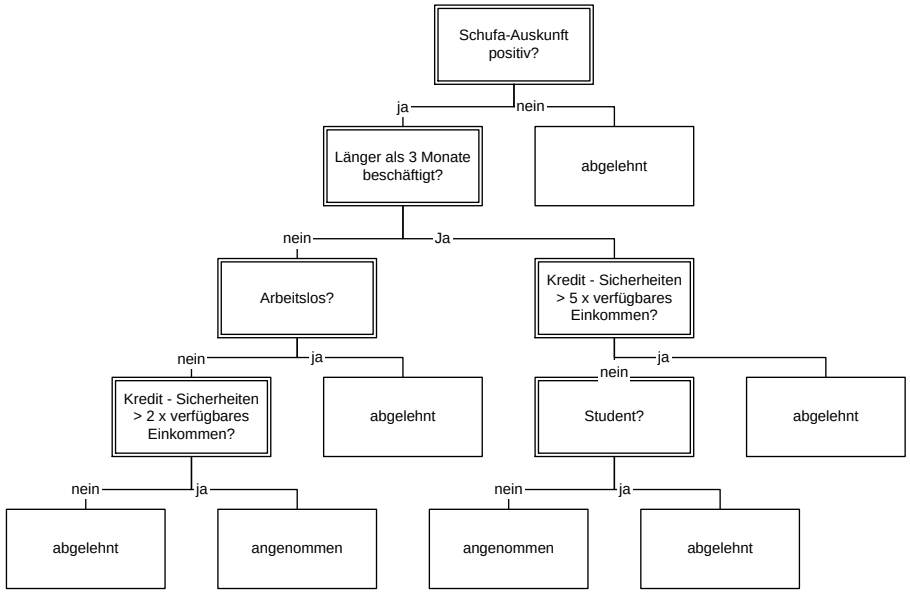
- Lernproblem: kontinuierliche Trainingsdaten.
  - $L = \langle (\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N) \rangle$
  - z.B. $\langle (\mathbf{x}_1, 3.5), \ldots, (\mathbf{x}_N, -2.8) \rangle$

Andere Lernprobleme

- Im Laufe des Semesters werden wir noch andere Problemstellungen kennen lernen.
- Z.B. ordinale Regression.
  - Präferenzlernen, Instanzen in die richtige Reihenfolge bringen.
- Z.B. Sequenzlernen.
  - Eingabe: Sequenz (z.B. Folge von Wörtern)
  - Ausgabe: Sequenz (z.B. Folge semantischer Tags).
- Z.B. Strukturlernen.
  - Eingabe: Sequenz, Baum, Graph (z.B. Text, Web).
  - Ausgabe: Baum, Graph (z.B. Parsbaum, Klassifikation von Webseiten).

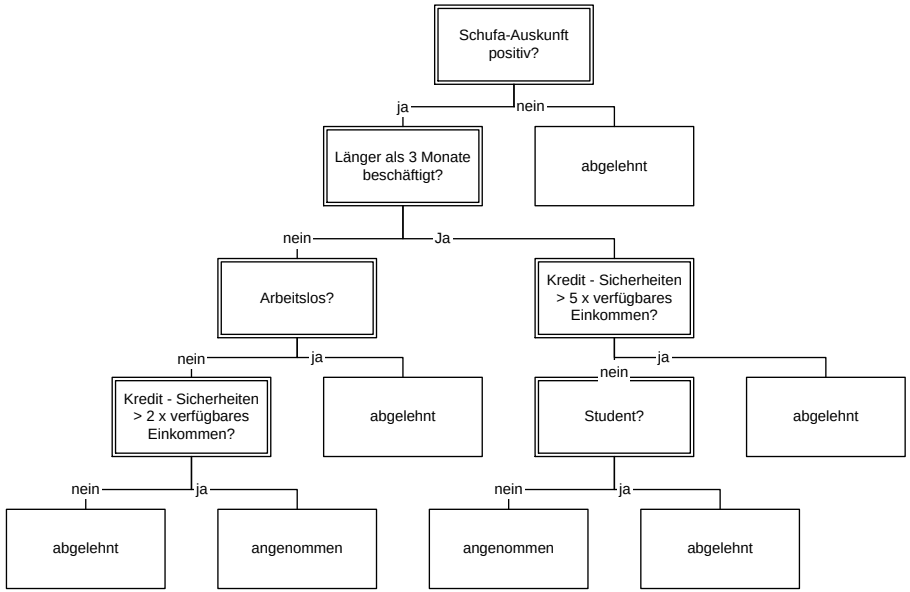
Entscheidungsbäume

- Testknoten: Testen, ob der Wert eines Attributs eine Bedingung erfüllen; bedingte Verzweigung in einen Ast.
- Terminalknoten: Liefern einen Wert als Ausgabe.



Entscheidungsbäume

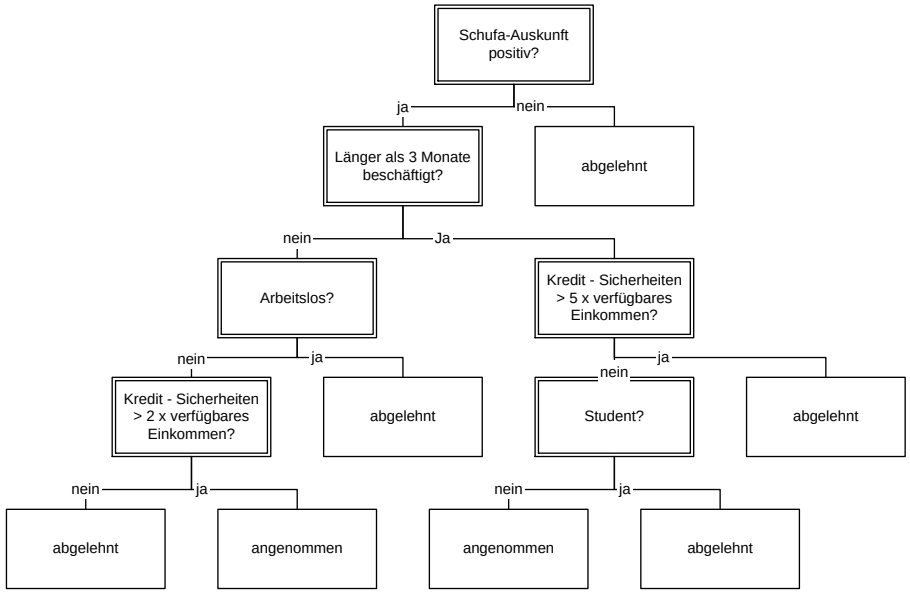
- Entscheidungsbäume können logische Operationen repräsentieren, wie:
  - $\wedge, \vee, XOR$
  - $(A \wedge B) \vee (C \wedge \neg D \wedge E)$



Anwendung von Entscheidungsbäumen

- Testknoten: führe Test aus, wähle passende Verzweigung, rekursiver Aufruf.

- Terminalknoten: liefere Wert als Klasse zurück.



Anwendung von Entscheidungsbäumen

- Entscheidungsbäume sinnvoll, wenn:
  - Instanzen durch Attribut-Werte-Paare beschrieben werden.
  - Zielattribut hat einen diskreten Wertebereicht.
    - Bei Erweiterung auf Modellbäume auch kontinuierlicherer Wertebereich möglich.
  - Interpretierbarkeit der Vorhersage gewünscht ist.

- Anwendungsgebiete:
  - Medizinische Diagnose
  - Kreditwürdigkeit

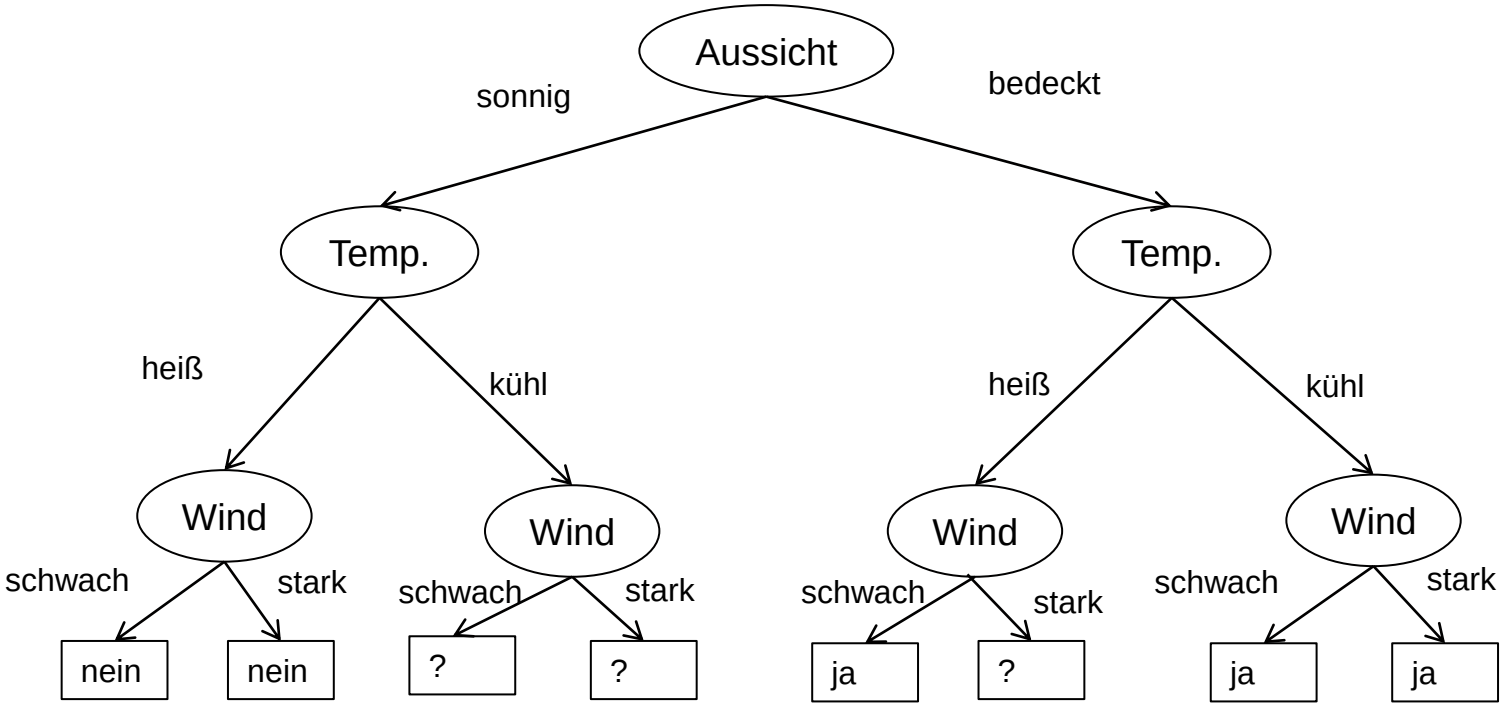
Lernen von Entscheidungsbäumen

Tag	Aussicht	Temperatur	Luftfeuchtigkeit	Wind	Tennis?
1	sonnig	heiß	hoch	schwach	nein
2	sonnig	heiß	hoch	stark	nein
3	bedeckt	heiß	hoch	schwach	ja
4	Regen	mild	hoch	schwach	ja
5	Regen	kühl	normal	schwach	ja
6	Regen	kühl	normal	stark	nein
7	bedeckt	kühl	normal	stark	ja
8	sonnig	mild	hoch	schwach	nein
9	sonnig	kühl	normal	schwach	ja
10	Regen	mild	normal	schwach	ja

- Finde Entscheidungsbaum, der zumindest für die Trainingsdaten die richtige Klasse liefert.
- Trivialer Weg: Erzeuge einen Baum, der lediglich die Trainingsdaten reproduziert.

Lernen von Entscheidungsbäumen

Tag	Aussicht	Temperatur	Wind	Tennis?
1	sonnig	heiß	schwach	nein
2	sonnig	heiß	stark	nein
3	bedeckt	heiß	schwach	ja
4	bedeckt	kühl	stark	ja
5	bedeckt	kühl	schwach	ja



Lernen von Entscheidungsbäumen

- Eleganter Weg: Unter den Bäumen, die mit den Trainingsdaten konsistent sind, wähle einen möglichst kleinen Baum (möglichst wenige Knoten).
- Kleine Bäume sind gut, weil:
  - sie leichter zu interpretieren sind;
  - sie in vielen Fällen besser generalisieren.
  - Es gibt mehr Beispiele pro Blattknoten. Die Klassenentscheidungen in den Blättern stützen sich so auf mehr Beispiele.

Vollständige Suche nach kleinstem Baum

- Wie viele Entscheidungsbäume gibt es?
  - Angenommen m binäre Attribute, zwei Klassen.

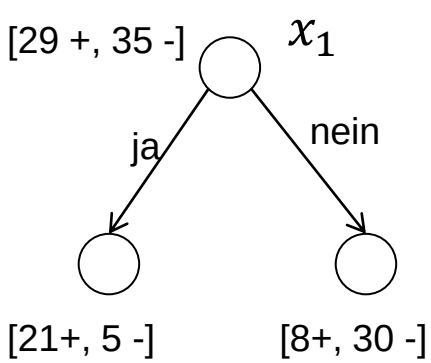
- Komplexität einer vollständigen Suche nach kleinstem Baum?

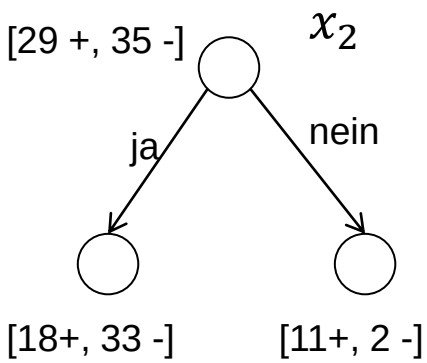
Lernen von Entscheidungsbäumen

- Greedy-Algorithmus, der einen kleinen Baum (statt des kleinsten Baumes) findet, dafür aber polynomiell in Anzahl der Attribute ist.
- Idee für Algorithmus?

Top-Down Konstruktion

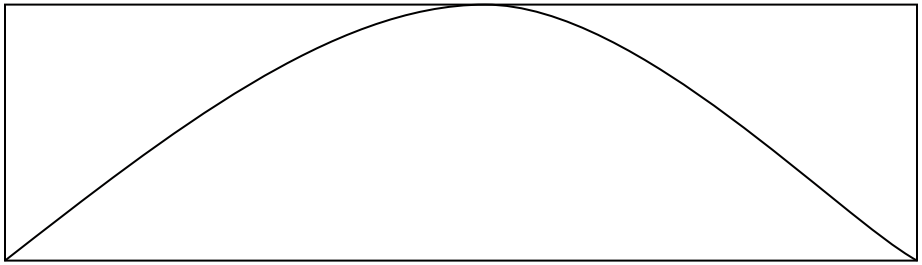
- Solange die Trainingsdaten nicht perfekt klassifiziert sind:
  - Wähle das „beste“ Attribut A, um die Trainingsdaten zu separieren.
  - Erzeuge für jeden Wert des Attributs A einen neuen Nachfolgeknoten und weise die Trainingsdaten dem so entstandenen Baum zu.
- Problem: Welches Attribut ist das beste?





Entropie

- Entropie = Maß für Unsicherheit.



- S ist Menge von Trainingsdaten.
- p_+ ist Anteil von positiven Beispielen in S.
- p_- ist Anteil von negativen Beispielen in S.
- Entropie misst die Unsicherheit in S:
  - $H(S) = -p_+ \log_2 p_+ \; - p_- \log_2 p_-$

Entropie

- $H(S) =$ Erwartete Anzahl von Bits, die benötigt werden, um die Zielklasse von zufällig gezogenen Beispielen aus der Menge S zu kodieren.
  - Optimaler, kürzester Kode

- Entropie einer Zufallsvariable y:
  - $H(y) = -\sum_v p(y = v) \log_2 p(y = v)$
- Empirische Entropie Zufallsvariable y auf Daten L:
  - $H(L, y) = -\sum_v \hat{p}_L(y = v) \log_2 \hat{p}_L(y = v)$

Bedingte Entropie

- Bedingte Entropie von Zufallsvariable y gegeben Ereignis $x=v$:
  - $H_{|x=v}(y) = H(y \mid x=v) = -\sum_{v'} p(y=v' \mid x=v) \log_2 p(y=v' \mid x=v)$

- Empirische bedingte Entropie auf Daten L:
  - $$H_{|x=v}(L,y) = H(L,y \mid x=v)$$
    $$= -\sum_{v'} \hat{p}_L(y=v' \mid x=v) \log_2 \hat{p}_L(y=v' \mid x=v)$$

Information Gain

- Entropie der Klassenvariable: Unsicherheit über korrekte Klassifikation.
- Transinformation eines Attributes („Information Gain“).
  - Verringerung der Entropie der Klassenvariable nach Test des Wertes des Attributes.
  - $IG(x) = H(y) - \sum_v p(x=v) H_{|x=v}(y)$
- Empirische Transinformation auf Daten L:
  - $IG(L,x) = H(L,y) - \sum_v \hat{p}_L(x=v) H(L_{|x=v}, y)$

Beispiel

IG = 0,05 | IG = 0,46 | IG = 0,05

Beispiele	x_1: Kredit > 3 x Einkommen?	x_2: Länger als 3 Monate beschäftigt?	x_3: Student?	y: Kredit zurückgezahlt?
1	Ja	Ja	Nein	Ja
2	Ja	Nein	Nein	Nein
3	Nein	Ja	Ja	Ja
4	Nein	Nein	Ja	Nein
5	Nein	Ja	Nein	Ja
6	Nein	Nein	Nein	Ja

$$H(L,y) = -\sum_v \hat{p}_L(y=v)\log_2 \hat{p}_L(y=v)$$

≈0.92

$$IG(L,x) = H(L,y) - \sum_v \hat{p}_L(x=v)H(L_{|x=v},y)$$

Beispiel II

- $IG(L,x)$= Erwartete Reduktion der Unsicherheit durch den Split an Attribut x.
- Welcher Split ist besser?

- $H(y,L) = -\left(\frac{29}{64}\log_2\frac{29}{64} + \frac{35}{64}\log_2\frac{35}{64}\right) = 0{,}99$
- $IG(L,x_1) = 0{,}99 - \left(\frac{26}{64} * H(L_{|x_1=ja},y) + \frac{38}{64} * H(L_{|x_1=nein},y)\right) = 0{,}26$
- $IG(L,x_2) = 0{,}99 - \left(\frac{51}{64} * H(L_{|x_2=ja},y) + \frac{13}{64} * H(L_{|x_2=nein},y)\right) = 0{,}11$

Information Gain / Info Gain Ratio

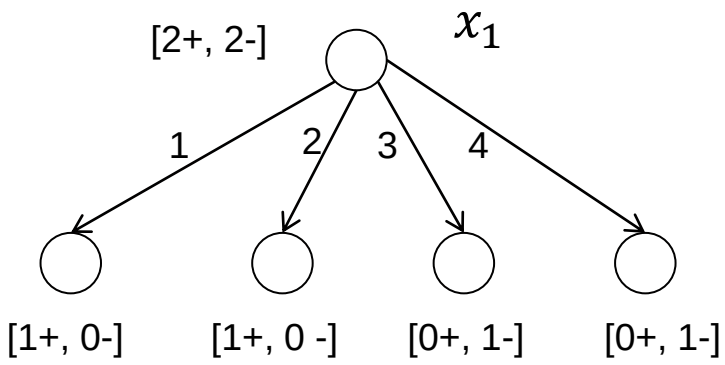
- Motivation:
  - Vorhersage ob ein Student die Prüfung besteht.
  - Wie hoch ist der Info Gain des Attributes „Matrikelnummer“?
  - Informationsgehalt des Tests ist riesig.

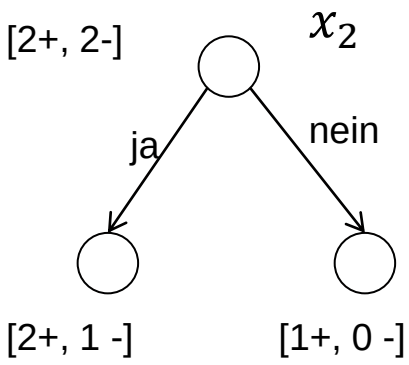
Information Gain / Info Gain Ratio

- Motivation:
  - Vorhersage ob ein Student die Prüfung besteht.
  - Wie hoch ist der Info Gain des Attributes „Matrikelnummer“?
  - Informationsgehalt des Tests ist riesig.
- Idee: Informationsgehalt des Tests bestrafen.
  - $$GainRatio(L,x) = \frac{IG(L,x)}{SplitInfo(L,x)}$$
  - $$SplitInfo(L,x) = -\sum_v \frac{|L_{|x=v}|}{|L|} \log_2 \frac{|L_{|x=v}|}{|L|}$$

Beispiel: Info Gain Ratio

- Welcher Split ist besser?





$IG(L, x_1) = 1$

$SplitInfo(L, x_1) = -4\left(\frac{1}{4}\log_2\frac{1}{4}\right) = 2$

$GainRatio(L, x_1) = 0{,}5$

$IG(L, x_2) = 1 - \left(\frac{3}{4} * 0{,}92\right) = 0{,}68$

$SplitInfo(L, x_2) = -\left(\frac{3}{4}\log_2\frac{3}{4} + \frac{1}{4}\log_2\frac{1}{4}\right) = 0{,}81$

$GainRatio(L, x_2) = 0{,}84$

Algorithmus ID3

- Voraussetzung:
  - Klassifikationslernen,
  - Alle Attribute haben festen, diskreten Wertebereich.
- Idee: rekursiver Algorithmus.
  - Wähle das Attribut, welches die Unsicherheit bzgl. der Zielklasse maximal verringert.
  - Dann rekursiver Aufruf für alle Werte des gewählten Attributs.
  - Solange, bis in einem Zweig nur noch Beispiele derselben Klasse sind.
- Originalreferenz:

  J.R. Quinlan: „Induction of Decision Trees". 1986

Algorithmus ID3

- Eingabe: $L = \langle (\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N) \rangle$, verfügbar= $(x_1, \ldots, x_m)$
- Wenn alle Bsp in L dieselbe Klasse y haben,
  - dann gib Terminalknoten mit Klasse y zurück.
- Wenn verfügbar=leer,
  - Gib Terminalknoten mit häufigster Klasse in L zurück.
- Sonst konstruiere Testknoten:
  - Bestimme bestes Attribut $x_* = \arg\max_{x_i \in \text{verfügbar}} IG(L, x_i)$
  - Für alle Werte v_j dieses Attributs:
    - Teile Trainingsmenge, $L_j = \langle (\mathbf{x}_k, y_k) \in L \mid x_{k*} = v_j \rangle$
    - Rekursion: Zweig für Wert v_j = ID3 (L_j , $\text{verfügbar} \setminus x_*$).

Algorithmus ID3: Beispiel

Beispiele	x_1: Kredit > 3 x Einkommen?	x_2: Länger als 3 Monate beschäftigt?	x_3: Student?	y: Kredit zurückgezahlt?
1	Ja	Ja	Nein	Ja
2	Ja	Nein	Nein	Nein
3	Nein	Ja	Ja	Ja
4	Nein	Nein	Ja	Nein
5	Nein	Ja	Nein	Ja
6	Nein	Nein	Nein	Ja

- Eingabe: $L = \langle (\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N) \rangle$, verfügbar= $(x_1, \ldots, x_m)$
- Wenn alle Bsp in L dieselbe Klasse y haben,
  - dann gib Terminalknoten mit Klasse y zurück.
- Wenn verfügbar=leer,
  - Gib Terminalknoten mit häufigster Klasse in L zurück.
- Sonst konstruiere Testknoten:
  - Bestimme bestes Attribut $x_* = \arg\max_{x_i \in \text{verfügbar}} IG(L, x_i)$
  - Für alle Werte v_j dieses Attributs:
    - Teile Trainingsmenge, $L_j = \langle (\mathbf{x}_k, y_k) \in L \mid x_{k*} = v_j \rangle$
    - Rekursion: Zweig für Wert v_j = ID3 (L_j , $\text{verfügbar} \setminus x_*$).

Kontinuierliche Attribute

- ID3 wählt Attribute mit größtem Informationsgehalt aus und bildet dann einen Zweig für jeden Wert dieses Attributes.
- Geht nur für diskrete Attribute.
- Attribute wie Größe, Einkommen, Entfernung haben unendlich viele Werte.

- Idee?

Kontinuierliche Attribute

- Idee: Binäre Entscheidungsbäume, „≤“-Tests.
- Problem: unendlich viele Werte für binäre Tests
- Idee: nur endlich viele Werte kommen in Trainingsdaten vor.
- Beispiel:
  - Kontinuierliches Attribut hat folgende Werte:
    - 0,2; 0,4; 0,7; 0,9
  - Mögliche Splits:
    - $\leq 0{,}2; \leq 0{,}4; \leq 0{,}7; \leq 0{,}9$
- Andere Möglichkeiten:
  - Attribute in einen diskreten Wertebereich abbilden.

Algorithmus C4.5

- Weiterentwicklung von ID3
- Verbesserungen:
  - auch kontinuierliche Attribute
  - behandelt Trainingsdaten mit fehlenden Attributwerten
  - behandelt Attribute mit Kosten
  - Pruning
- Nicht der letzte Stand: siehe C5.0

- Originalreferenz:

  J.R. Quinlan: „ C4.5: Programs for Machine Learning". 1993

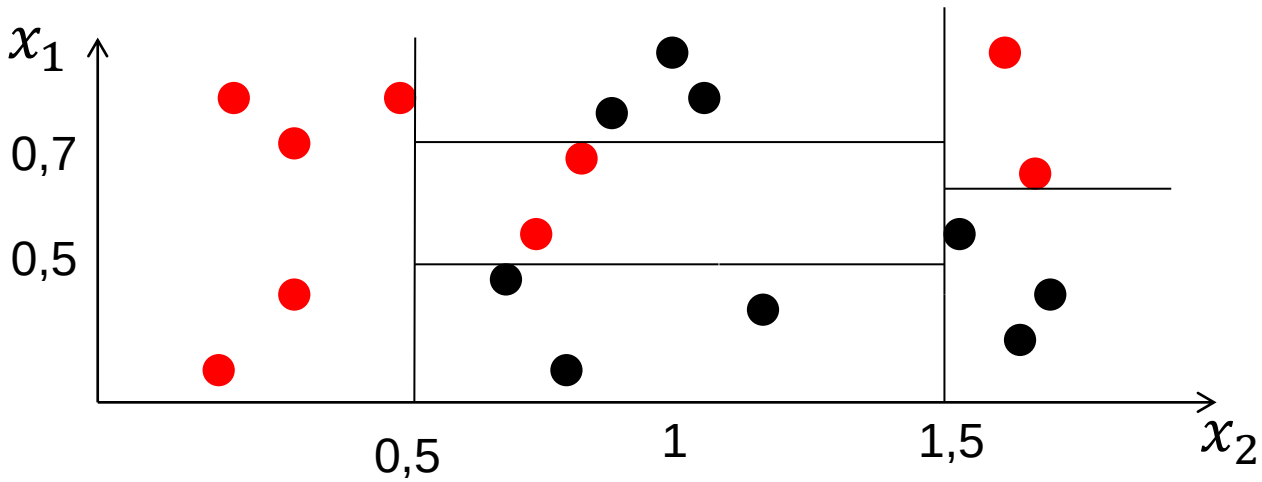
Algorithmus C4.5

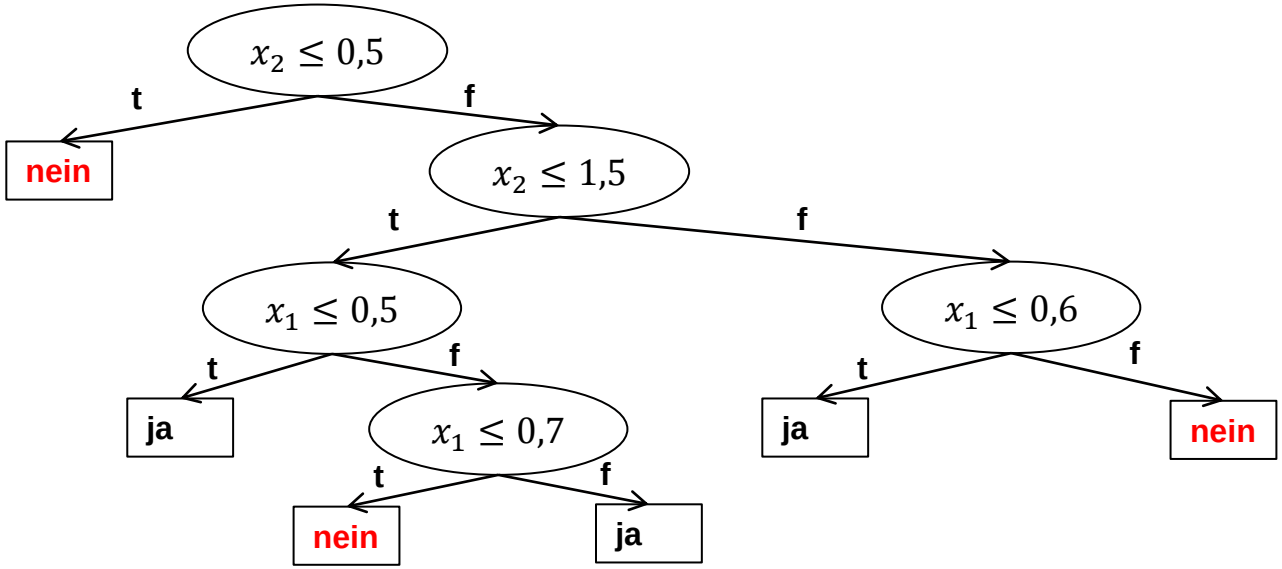
- Eingabe: $L = \langle (\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N) \rangle$.
- Wenn alle Bsp in L dieselbe Klasse y haben,
  - dann gib Terminalknoten mit Klasse y zurück.
- Wenn alle Instanzen identisch sind:
  - Gib Terminalknoten mit häufigster Klasse in L zurück.
- Sonst konstruiere besten Testknoten, iteriere dazu über alle Attribute.
  - Diskrete Attribute: wie ID3.
  - Kontinuierliche Attribute: $[x_* \leq v_*] = \arg\max_{\text{Attribute } x_i,\ \text{Werte } v \text{ in } L} IG(L, [x_i \leq v])$
  - Wenn bestes Attribut diskret, Rekursion wie ID3.
  - Wenn bestes Attribut kontinuierlich, teile Trainingsmenge:
    - $L_{links} = \langle (\mathbf{x}_k, y_k) \in L \mid x_{k*} \leq v_* \rangle$, $L_{rechts} = \langle (\mathbf{x}_k, y_k) \in L \mid x_{k*} > v_* \rangle$
    - Rekursion: linker Zweig= $C4.5(L_{links})$, rechter Zweig= $C4.5(L_{rechts})$

Information Gain für kontinuierliche Attribute

- Information Gain eines Tests „ $[x \leq v]$ “:
  - $IG([x \leq v]) = H(y) - p([x \leq v])H_{[x \leq v]}(y) - p([x > v])H_{[x > v]}(y)$
- Empirischer Information Gain:
  - $IG(L,[x \leq v]) = H(L,y) - \hat{p}_L([x \leq v])H(L_{\|[x \leq v]},y) - \hat{p}_L([x > v])H(L_{\|[x > v]},y)$

C4.5: Beispiel





Pruning

- Problem: Blattknoten, die nur von einem (oder sehr wenigen) Beispielen gestützt werden, liefern häufig keine gute Klassifikation.
- Pruning: Entfernen von Testknoten, die Blätter mit weniger als einer Mindestzahl von Beispielen erzeugen.
- Dadurch entstehen Blattknoten, die dann mit der am häufigsten auftretenden Klasse beschriftet werden.

Pruning mit Schwellwert

- Für alle Blattknoten: Wenn weniger als r Trainingsbeispiele in den Blattknoten fallen
  - Entferne darüberliegenden Testknoten.
  - Erzeuge neuen Blattknoten, sage Mehrheitsklasse vorher.

- Regularisierungsparameter r.
- Einstellung mit Cross Validation.

Reduced Error Pruning

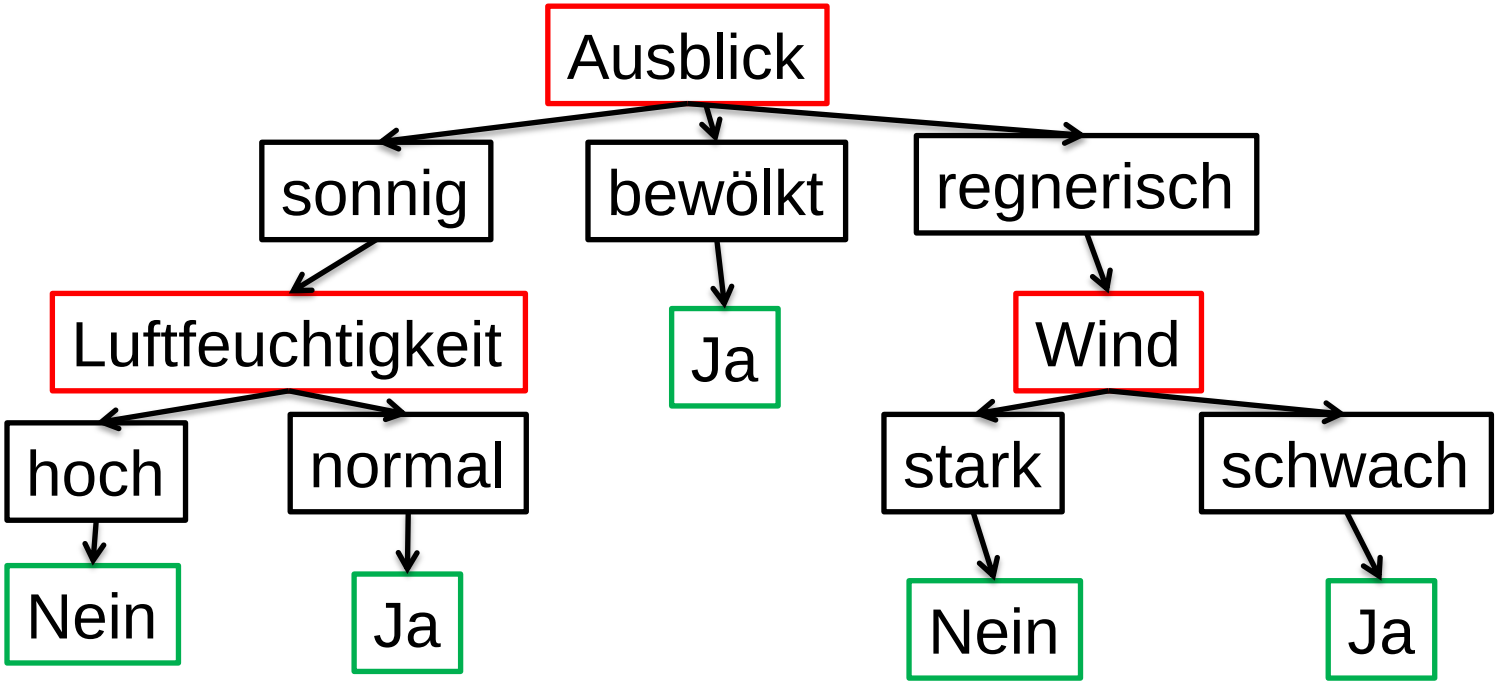
- Aufteilen der Trainingsdaten in Trainingsmenge und Pruningmenge.
- Nach dem Aufbau des Baumes mit der Trainingsmenge:
  - Versuche, zwei Blattknoten durch Löschen eines Tests zusammenzulegen,
  - Solange dadurch die Fehlerrate auf der Pruningmenge verringert wird.

Umwandlung von Bäumen in Regeln

- Pfad durch den Baum: Bedingung der Regel
- Klasse: Schlussfolgerung

- Pruning von Regeln: Probiere aus, welche Bedingungen weggelassen werden können, ohne dass die Fehlerrate dadurch steigt.

Umwandlung von Bäumen in Regeln



$R_1 = Wenn\ (Ausblick = sonnig) \wedge (Luftfeuchtigkeit = hoch), dann\ kein\ Tennis\ spielen$

$R_2 = Wenn\ (Ausblick = sonnig) \wedge (Luftfeuchtigkeit = normal), dann\ Tennis\ spielen$

$R_3 = Wenn\ (Ausblick = bewölkt), dann\ Tennis\ spielen$

$R_4 = Wenn\ (Ausblick = regnerisch) \wedge (Wind = stark), dann\ kein\ Tennis\ spielen$

$R_6 = Wenn\ (Ausblick = regnerisch) \wedge (Wind = schwach), dann\ Tennis\ spielen$

Induktiver Bias von Entscheidungsbäumen

- Version Space: Raum aller Hypothesen, die mit den Daten konsistent sind.
- Hypothesen im Version Space unterscheiden sich in ihrer Klassifikation neuer Daten.
- Welche Hypothese ist die Richtige?
- Induktiver Bias: Bevorzugung einer bestimmten Hypothese im Version Space.
- Jeder Lernalgorithmus, der nur eine Hypothese zurückliefert muss einen induktiven Bias haben.
- Induktiver Bias von ID3: kleine Bäume werden bevorzugt.

Entscheidungsbäume aus großen Datenbanken: SLIQ

- C4.5 iteriert häufig über die Trainingsmenge (wie häufig?)
- Wenn die Trainingsmenge nicht in den Hauptspeicher passt, wird das Swapping unpraktikabel!
- SLIQ:
  - Vorsortieren der Werte für jedes Attribut
  - Baum „breadth-first“ aufbauen, nicht „depth-first“.
- Originalreferenz:

  M. Mehta et. al.: „SLIQ: A Fast Scalable Classifier for Data Mining”. 1996

SLIQ: Gini Index

- Um besten Split zu bestimmen, verwendet SLIQ nicht den Information Gain, sondern den Gini-Index.
- Für eine Trainingsmenge L mit n unterschiedlichen Klassen:
  - $Gini(L) = 1 - \sum_{j=1\ldots n} p_j^2$
- Nach einem binären Split der Menge L in die Mengen L_1und L_2 ergibt sich:
  - $Gini_{split}(L) = \frac{|L_1|}{|L|} Gini(L_1) + \frac{|L_2|}{L_2} Gini(L_2)$
- Gini-Index verhält sich ähnlich wie Information Gain.

SLIQ - Algorithmus

1. Starte **Vorsortieren** der Beispiele
2. Solange Abbruchkriterium noch nicht erreicht ist
   1. Für alle Attribute
      1. Lege für alle Knoten ein Klassenhistogramm an.
      2. Starte **Evaluierung** der Splits
   2. Wähle Split
   3. Aktualisiere Entscheidungsbaum; für jeden neuen Knoten **aktualisiere Klassenliste** (Knoten)

SLIQ (1. Teilschritt)

- Vorsortieren der Beispiele

1. Für jedes Attribut: Lege eine *Attributliste* mit den Spalten Wert, Beispiel-ID und Klasse an.
2. Lege eine *Klassenliste* mit den Spalten Beispiel-ID, Klasse und Knoten an.
3. Iteriere über alle Trainingsbeispiele
   - Für alle Attribute
     - Füge Attributwert, Beispiel-ID und Klasse sortiert (nach Attributwert) in alle Listen ein.
     - Füge die Beispiel-ID, die Klasse und den Knoten (1) nach Beispiel-ID sortiert in die Klassenliste ein.

SLIQ: Beispiel

TRAINING DATA

Age	Salary	Class
30	65	G
23	15	B
40	75	G
55	40	B
55	100	G
45	60	G

AFTER PRE-SORTING

Age List

Age	Class List Index

Salary List

Salary	Class List Index

Class List

	Class	Leaf
1	G	
2		
3		
4		
5		
6		

SLIQ: Beispiel

TRAINING DATA

Age	Salary	Class
30	65	G
23	15	B
40	75	G
55	40	B
55	100	G
45	60	G

AFTER PRE-SORTING

Age List

Age	Class List Index
23	2
30	1
40	3
45	6
55	5
55	4

Salary List

Salary	Class List Index
15	2
40	4
60	6
65	1
75	3
100	5

Class List

	Class	Leaf
1	G	N1
2	B	N1
3	G	N1
4	B	N1
5	G	N1
6	G	N1

SLIQ

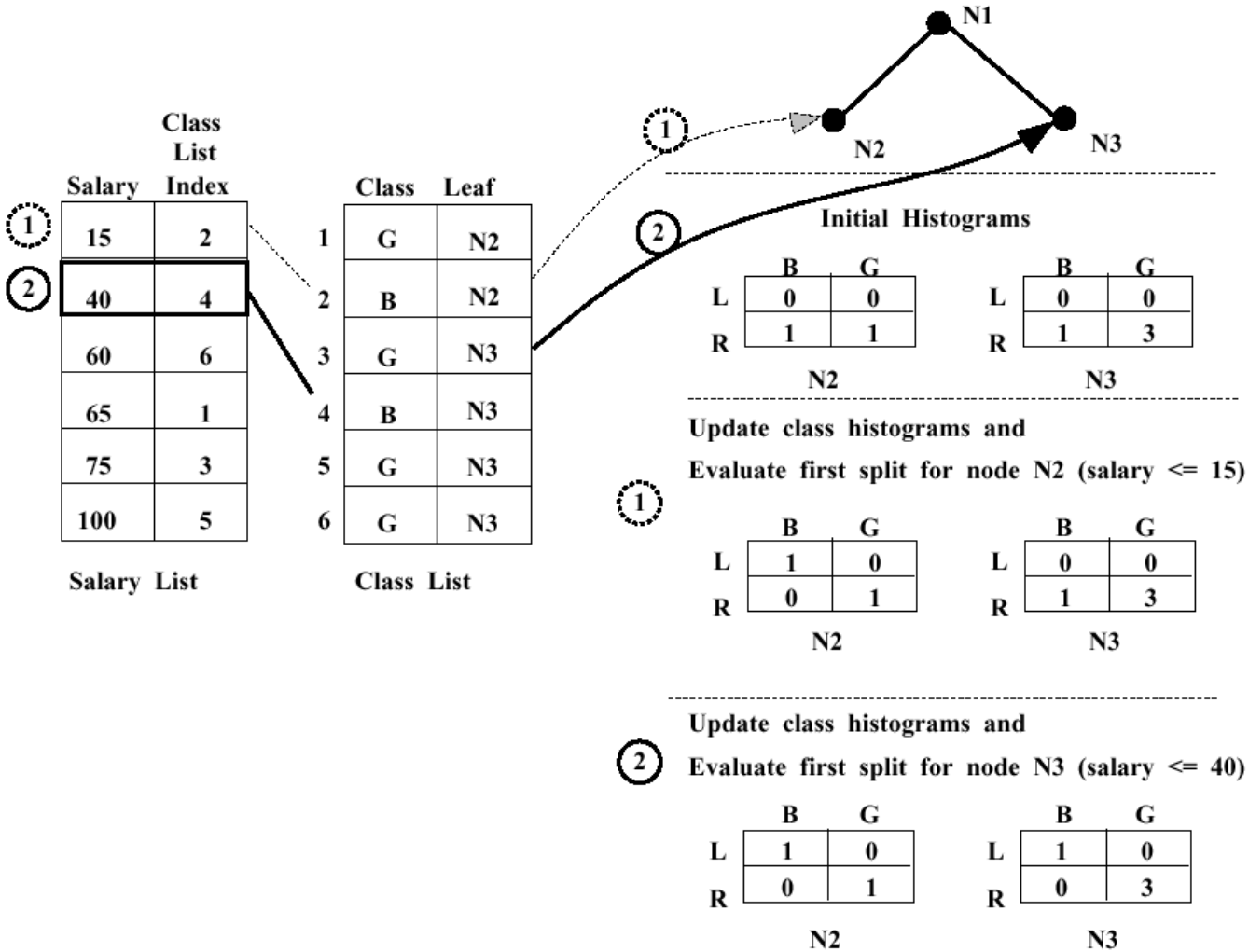
1. Starte **Vorsortieren** der Beispiele ✓
2. Solange Abbruchkriterium noch nicht erreicht ist
   1. Für alle Attribute
      1. Lege für alle Knoten ein Klassenhistogramm an.
      2. Starte **Evaluierung** der Splits
   2. Wähle Split
   3. Aktualisiere Entscheidungsbaum; für jeden neuen Knoten **aktualisiere Klassenliste** (Knoten)

SLIQ (2. Teilschritt)

- Evaluiere Splits

1. Für alle Knoten, für alle Attribute
   1. Lege ein Histogramm an (für alle Klassen speichert das Histogramm die Anzahl der Beispiele vor und nach dem Split).
2. Für alle Attribute *x*
   1. Für jeden Wert v (traversiere Attributliste für A)
      1. Finde entsprechenden Eintrag in der Klassenliste (liefert Klasse und Knoten).
      2. Aktualisiere Histogramm für den Knoten.
      3. Bewerte den Split (falls Maximum, merken!)

SLIQ: Beispiel



SLIQ

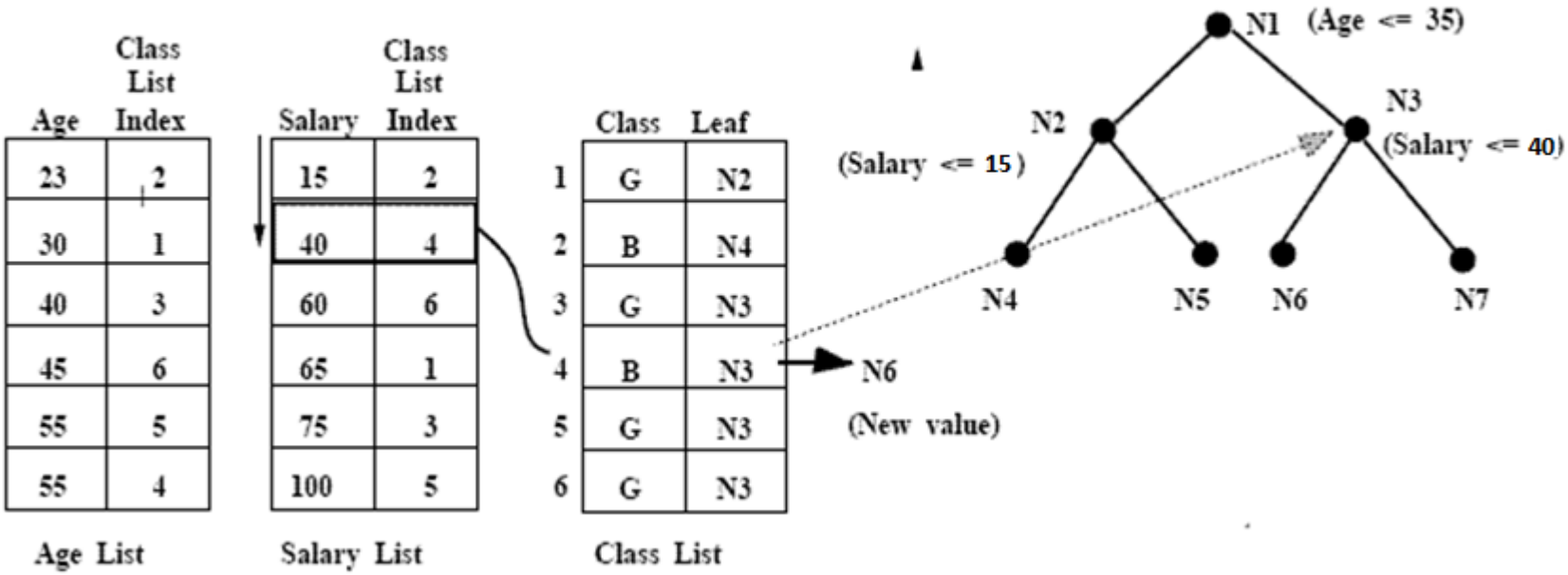
1. Starte **Vorsortieren** der Beispiele ✓
2. Solange Abbruchkriterium noch nicht erreicht ist
   1. Für alle Attribute
      1. Lege für alle Knoten ein Klassenhistogramm an.
      2. Starte **Evaluierung** der Splits ✓
   2. Wähle Split
   3. Aktualisiere Entscheidungsbaum; für jeden neuen Knoten **aktualisiere Klassenliste** (Knoten)

SLIQ (3. Teilschritt)

- Aktualisiere Klassenliste (Knoten)

1. Traversiere Attributliste des im Knoten verwendeten Attributs.
2. Für alle Einträge (Wert, ID)
3. Finde passenden Eintrag (ID, Klasse, Knoten) in der Klassenliste.
4. Wende Split-Kriterium an, ermittle neuen Knoten.
5. Ersetze Klassenlisteneintrag durch (ID, Klasse, neuer Knoten).

SLIQ: Beispiel



SLIQ: Datenstrukturen

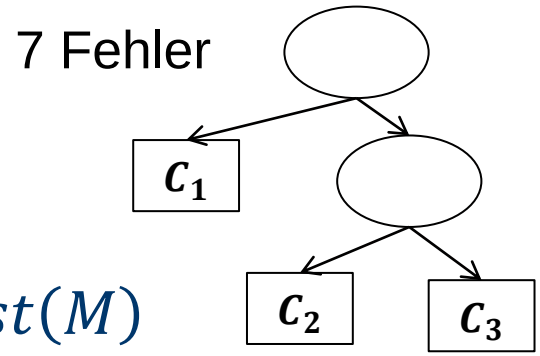
- Datenstrukturen im Speicher
- Ausgelagerte Datenstrukturen
- Datenstrukturen in Datenbank

SLIQ- Pruning

- Minimum Description Length (MDL): Bestes Modell für einen gegebenen Datensatz minimiert Summe aus Länge der im Modell kodierten Daten plus Länge des Modells.
  - $cost(M, D) = cost(D|M) + cost(M)$
- $cost(M)$ = Kosten des Modells (Länge).
  - Wie groß ist der Entscheidungsbaum.
- $cost(D|M)$ = Kosten, die Daten durch das Modell zu beschreiben.
  - Wie viele Klassifikationsfehler entstehen.

Pruning – MDL Beispiel I

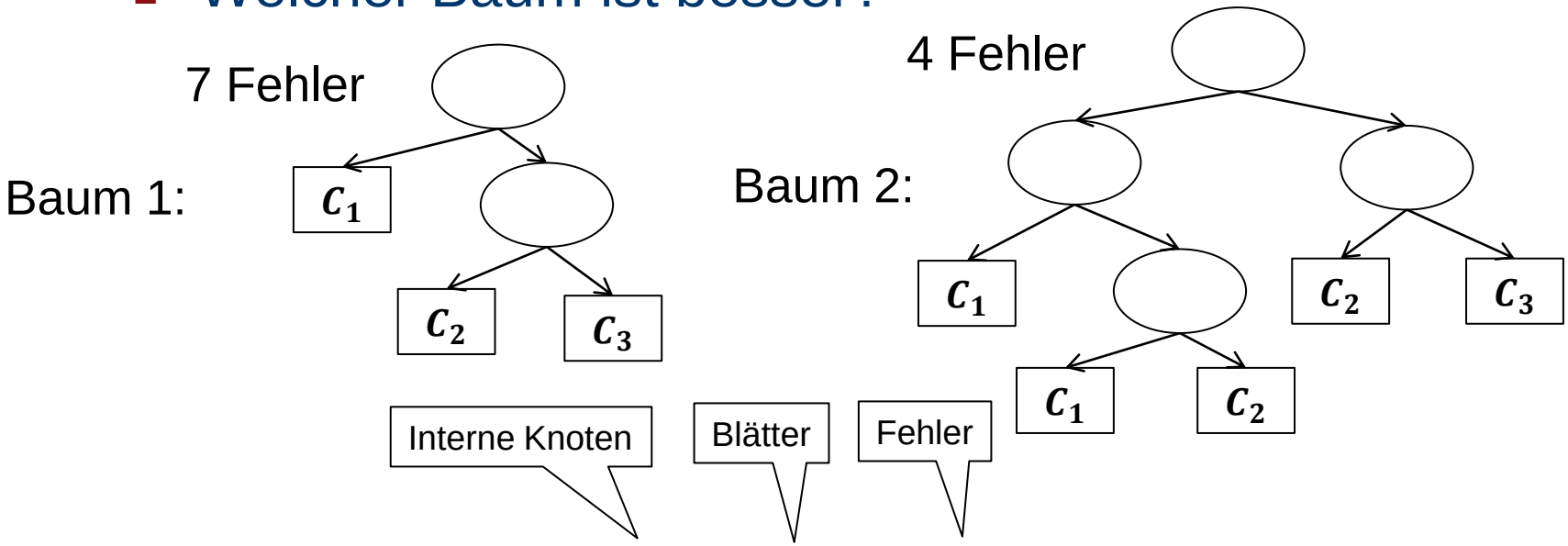
- Angenommen: 16 binäre Attribute und 3 Klassen



- $cost(M, D) = cost(D|M) + cost(M)$
  - Kosten für die Kodierung eines internen Knotens:
    - $\log_2(m) = \log_2(16) = 4$
  - Kosten für die Kodierung eines Blattes:
    - $\lceil \log_2(k) \rceil = \lceil \log_2(3) \rceil = 2$
  - Kosten für die Kodierung eines Klassifikationsfehlers bei n Trainingsdaten:
    - $\log_2(n)$

Pruning – MDL Beispiel II

- Welcher Baum ist besser?



- Kosten für Baum 1: $2 * 4 + 3 * 2 + 7 * \log_2 n = 14 + 7\log_2 n$
- Kosten für Baum 2: $4 * 4 + 5 * 2 + 4 * \log_2 n = 26 + 4\log_2 n$
- Wenn $n < 16$, dann ist Baum 1 besser.
- Wenn $n > 16$, dann ist Baum 2 besser.

SLIQ – Eigenschaften

- Laufzeit für die Initialisierung (Vorsortierung):
  - $O(n \log n)$ für jedes Attribut
- Großteil der Daten muss nicht im Hauptspeicher gehalten werden.
- Gute Skalierbarkeit.
  - Wenn Klassenliste nicht mehr in Hauptspeicher passt funktioniert SLIQ nicht mehr.
  - Alternative: SPRINT
- Kann numerische und diskrete Attribute verarbeiten.
- Parallelisierbarkeit des Verfahrens.

Regressionsbäume

- ID3, C4.5, SLIQ: Lösen Klassifikationsproblem.
  - Ziel: niedrige Fehlerrate + kleine Bäume
- Attribute können kontinuierlich sein (außer bei ID3), aber Vorhersage ist diskret.
- Regression: Vorhersage kontinuierlicher Werte.
  - Ziel: niedrige Fehlerquadrate + einfaches Modell.
  - $SSE = \sum_{j=1}^{m} (y_j - f(x_j))^2$
- Methoden, die wir jetzt gleich ansehen:
  - Regressionsbäume,
  - Lineare Regression,
  - Modellbäume.

Regressionsbäume

- Eingabe: $L = \langle (\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N) \rangle$, kontinuierliche y.
- Gesucht: $f : X \to Y$
- Algorithmus CART
  - Wurde gleichzeitig, unabhängig von C4.5 entwickelt.
  - Terminalknoten enthalten kontinuierlichen Wert.
  - Algorithmus wie C4.5. Für Klassifikation leicht anderes Kriterium (Gini statt IG), kaum Unterschied.
  - Information Gain (und Gini) funktionieren nur für Klassifikation.
  - Split-Kriterium für Regression?
- Originalreferenz:

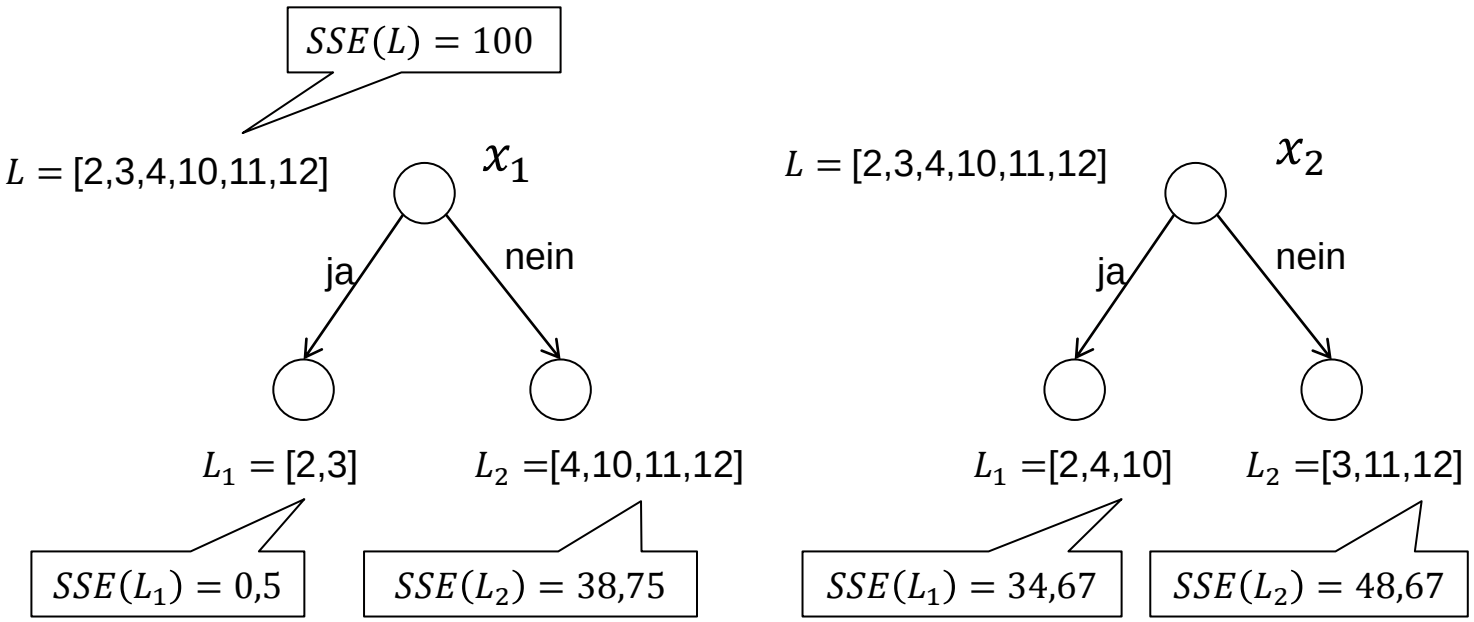
  L. Breiman et. al.: „Classification and Regression Trees". 1984

Regressionsbäume

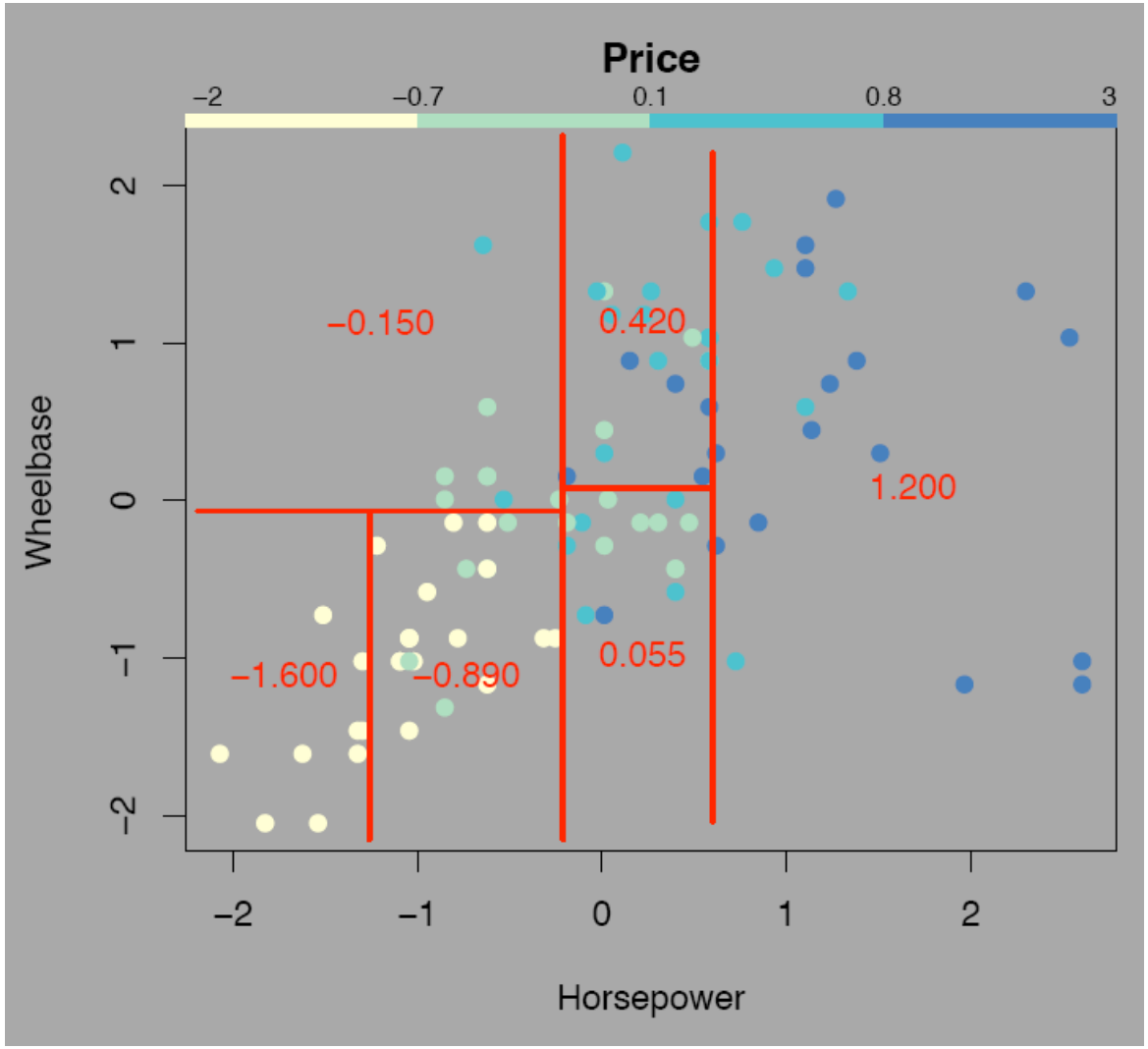
- Ziel: niedrige Fehlerquadrate (SSE) + kleiner Baum.
- Beispiele L kommen im aktuellen Knoten an.
- SSE im aktuellen Knoten: $SSE_L = \sum_{(x,y)\in L} (y-\bar{y})^2 \text{ mit } \bar{y} = \frac{1}{|L|}\sum_{(x,y)\in L} y$
- Mit welchem Test soll gesplittet werden?
- Kriterium für Testknoten $[x \le v]$:
  - $\text{SSE - Red}(L,[x \le v]) = SSE_L - SSE_{L|[x\le v]} - SSE_{L|[x>v]}$
  - SSE-Reduktion durch Test.
- Stop-Kriterium:
  - Führe keinen neuen Split ein, wenn SSE nicht um Mindestbetrag reduziert wird.
  - Erzeuge dann Terminalknoten mit Mittelwert m aus L.

CART- Beispiel

- Welcher Split ist besser?



Regressionsbäume



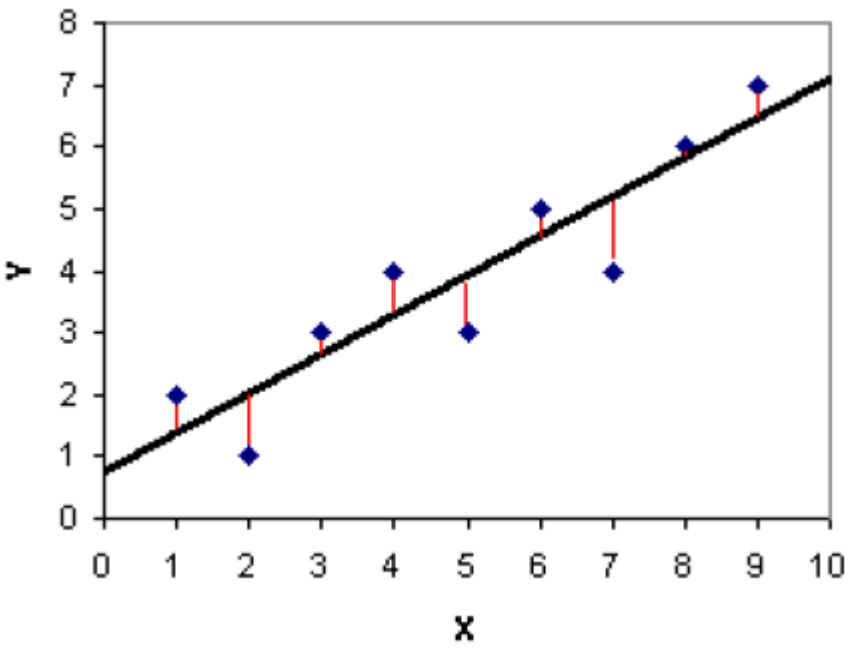
Lineare Regression

- Regressionsbäume: konstante Vorhersage in jedem Blattknoten.

- Lineare Regression: Globales Modell einer linearen Abhängigkeit zwischen **x** und *y*.
- Standardverfahren.
- Für sich genommen nützlich und Voraussetzung für Modellbäume.

Lineare Regression

- Eingabe: $L = \langle (\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m) \rangle$
- Gesucht: lineares Modell $f(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\mathbf{x} + c$



Normalenvektor

Residuen

Punkte auf der Regressionsgeraden stehen senkrecht auf Normalenvektor.

Lineare Regression

- Eingabe: $L = \langle (\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N) \rangle$
- Gesucht: lineares Modell $f(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\mathbf{x} + c$
- Vereinfachung zum besseren Verständnis der Herleitung: nur eine Dimension.
  - $f(x) = w^{\mathrm{T}}x + c$
- Gesucht: Parameter **w** und *c*, die SSE minimieren.
- SSE Ableiten, Ableitung auf null setzen (wie in der Schule).

Lineare Regression, eindimensional

- $SSE = \sum_{j=1}^{N} (y_j - f(\mathbf{x}_j))^2 = \sum_{j=1}^{N} (y_j - (w\mathbf{x}_j + c))^2$
- $\frac{\partial SSE}{\partial c} = \sum_{j=1}^{N} 2(y_j - w\mathbf{x}_j - c)(-1) = 0$

$$\boxed{(y_j - w\mathbf{x}_j - c)^2}$$

$$\Leftrightarrow \sum_{j=1}^{N} y_j - \sum_{j=1}^{N} w\mathbf{x}_j - Nc = 0$$

$$\Leftrightarrow c^* = \bar{y} - w\bar{\mathbf{x}}$$

Lineare Regression, eindimensional

- $$SSE = \sum_{j=1}^{N} (y_j - f(\mathbf{x}_j))^2 = \sum_{j=1}^{N} (y_j - (w\mathbf{x}_j + c))^2$$

- $$\frac{\partial SSE}{\partial w} = \sum_{j=1}^{N} \frac{\partial (y_j - w\mathbf{x}_j - c^*)^2}{\partial w}$$

$$\boxed{c^* = \bar{y} + w\bar{\mathbf{x}}}$$

$$= \sum_{j=1}^{N} \frac{\partial (y_j - w\mathbf{x}_j - \bar{y} + w\bar{\mathbf{x}})^2}{\partial w}$$

$$= \sum_{j=1}^{N} 2(y_j - w\mathbf{x}_j - \bar{y} + w\bar{\mathbf{x}})(-\mathbf{x}_j + \bar{\mathbf{x}}) = 0$$

$$\Leftrightarrow w^* = \frac{\sum_{j=1}^{N} (\bar{y} - y_j)(\bar{\mathbf{x}} - \mathbf{x}_j)}{\sum_{j=1}^{N} (\bar{\mathbf{x}} - \mathbf{x}_j)^2}$$

Lineare Regression

- Eingabe: $L = \langle (\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N) \rangle$
- Gesucht: lineares Modell $f(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\mathbf{x} + c$
- kleiner Trick:
  - $$f(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\mathbf{x} + c$$
    $$= (w_1 \quad \ldots \quad w_m) \begin{pmatrix} x_1 \\ \ldots \\ x_m \end{pmatrix} + c$$
    $$= (w_1 \quad \ldots \quad w_m \quad c) \begin{pmatrix} x_1 \\ \ldots \\ x_m \\ 1 \end{pmatrix} = \mathbf{w}'^{\mathrm{T}}\mathbf{x}'$$
- c kann weg, dafür eine zusätzliche Dimension.

Lineare Regression

- Fehlerquadrate:
  - $$SSE = \sum_{j=1}^{N} (y_j - f(\mathbf{x}_j))^2$$
    $$= \sum_{j=1}^{N} (y_j - (\mathbf{w}^{\mathrm{T}}\mathbf{x}_j))^2$$
    $$= (\mathbf{y} - \mathbf{X}\mathbf{w})^{\mathrm{T}}(\mathbf{y} - \mathbf{X}\mathbf{w})$$
- Gesucht: Parameter **w** und *c*, die SSE minimieren.
- Ableiten, Ableitung auf null setzen (wie in der Schule).
  - $$\frac{\partial SSE}{\partial \mathbf{w}} = \sum_{j=1}^{N} (y_j - (\mathbf{w}^{\mathrm{T}}\mathbf{x}_j))^2 = 0$$
    $$\Leftrightarrow \mathbf{w} = (\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y}$$
  - Inverse existiert nicht immer $\rightarrow$ Regularisierungsterm addieren (Ridge Regression).

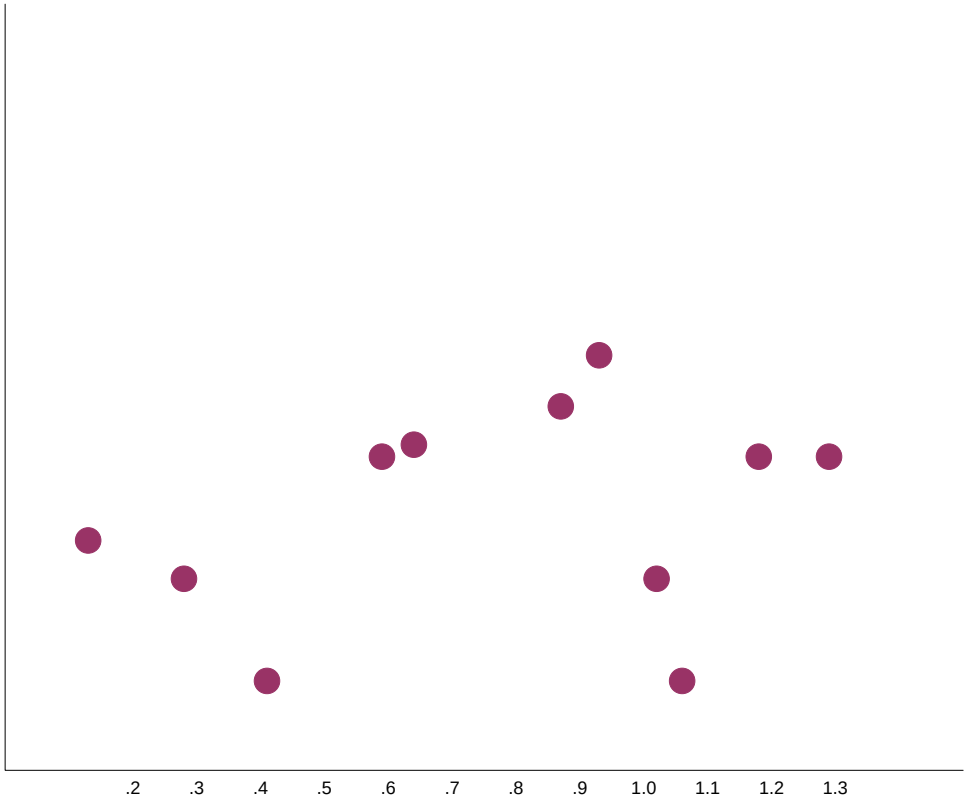
Ridge Regression

- Ziel: Summe aus Fehlerquadraten und Regularisierungsterm minimieren:
  - $w^* = \arg\min_{w} (\boldsymbol{y} - \mathbf{X}\mathbf{w})^T (\boldsymbol{y} - \mathbf{X}\mathbf{w}) + \lambda \mathbf{w}^T \mathbf{w}$
  - λ ist Regularisierungsparameter.

- Gesucht: Parameter **w**, der Summe aus Fehlerquadrate und Regularisierungsterm minimiert.
- Lösung: Ableiten und 0 setzen:
  - $w^* = (\boldsymbol{X}^T \boldsymbol{X} + \lambda \boldsymbol{I})^{-1} \boldsymbol{X}^T \boldsymbol{y}$

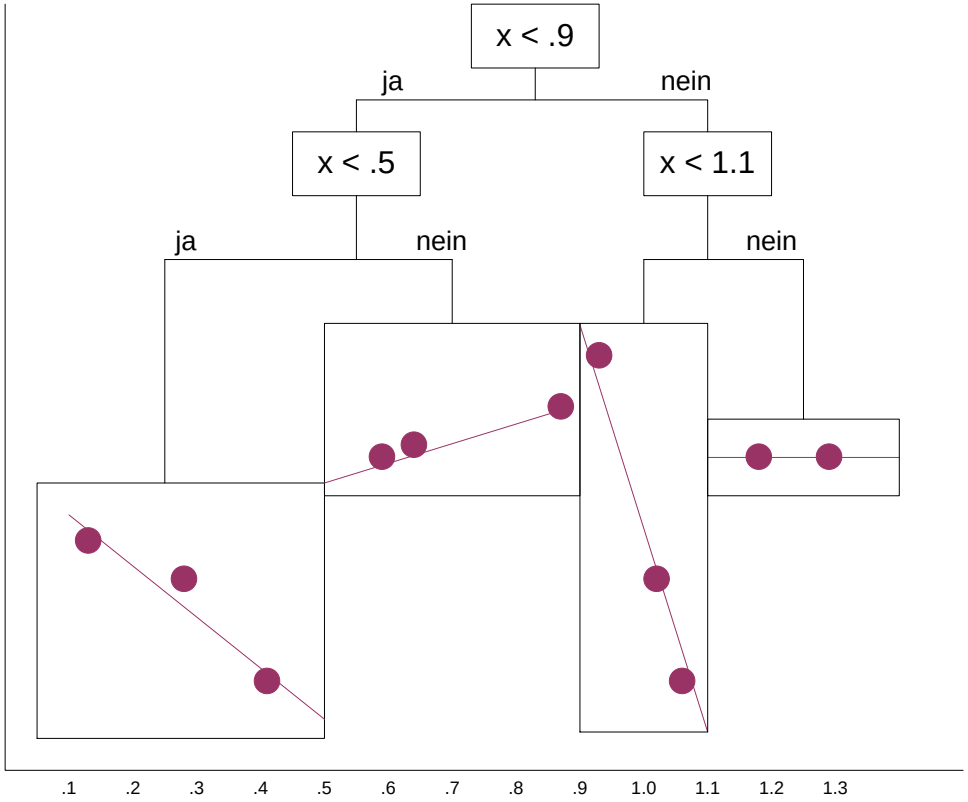
Modellbäume

- Entscheidungsbaum, aber lineares Regressionsmodell in Blattknoten.



Modellbäume

- Entscheidungsbaum, aber lineares Regressionsmodell in Blattknoten.



Modellbäume: Aufbau neuer Testknoten

- Führe lineare Regression über alle Beispiele aus, die den aktuellen Knoten erreichen.
- Bestimme SSE.
- Iteriere über alle möglichen Tests (wie C4.5).
  - Führe lineare Regressionen für Teilmengen der Beispiele im linken Zweig aus, bestimme SSE_{links}.
  - Führe lineare Regressionen für Teilmengen der Beispiele im rechten Zweig aus, bestimme SSE_{rechts}.
- Wähle Test mit größtem SSE- SSE_{links}- SSE_{rechts}.
- Stop-Kriterium: Wenn SSE nicht um Mindestbetrag reduziert wird, führe keinen neuen Test ein.

Fehlende Attributwerte

- Problem: Trainingsdaten mit fehlenden Attributwerten.
- Was, wenn Attribut A getestet wird, für das nicht für jedes Beispiel ein Attributwert existiert?
  - Trainingsbeispiel bekommt Attributwert, der unter allen Beispielen im Knoten n am häufigsten für A vorkommt.
  - Trainingsbeispiel bekommt Attributwert, den die meisten Beispiele mit der gleichen Klassifikation haben.

Attribute mit Kosten

- Beispiel: medizinische Diagnose
  - Bluttest kostet mehr als Pulsmessung
- Fragestellung: Wie kann man einen mit den Trainingsdaten konsistenten Baum mit geringen Kosten lernen?
- Antwort: Ersetzte Information Gain durch:
  - Tan and Schlimmer (1990): $\frac{IG^2(L,x)}{Cost(A)}$
  - Nunez (1988): $\frac{2^{IG\,(L,x)}-1}{(Cost(A)+1)^w}$
  - Parameter $w \in [0,1]$ steuert den Einfluss der Kosten.

Entscheidungsbaum - Vorteile

- Einfach zu interpretieren.
- Können effizient aus vielen Beispielen gelernt werden.
  - SLIQ, SPRINT
- Viele Anwendungen.
- Hohe Genauigkeit.

Entscheidungsbaum - Nachteile

- Nicht robust gegenüber Rauschen
- Tendenz zum Overfitting
- Instabil

Bootstrapping

- Einfache Resampling-Methode.
- Erzeugt viele Varianten einer Trainingsmenge L.
- Bootstrap Algorithmus:
  - Wiederhole M mal:
    - Generiere ein Sample L_k der Größe n von L:
      - Wiederholtes gleichverteiltes Ziehen mit Zurücklegen von Beispielen.
    - Berechne Modell θ_k für L_k
  - Gib die Bootstrap-Werte zurück:
    - $\theta^* = (\theta_1, \dots, \theta_k)$

Bootstrapping - Beispiel

$$x = \begin{pmatrix} 3{,}12 \\ 0 \\ 1{,}57 \\ 19{,}67 \\ 0{,}22 \\ 2{,}20 \end{pmatrix}, \bar{x} = 4{,}46$$

$$x = \begin{pmatrix} 1{,}57 \\ 0{,}22 \\ 19{,}67 \\ 0 \\ 0{,}22 \\ 3{,}12 \end{pmatrix}, \bar{x} = 4{,}13$$

$$x = \begin{pmatrix} 0 \\ 2{,}20 \\ 19{,}67 \\ 2{,}20 \\ 2{,}20 \\ 1.57 \end{pmatrix}, \bar{x} = 4{,}64$$

$$x = \begin{pmatrix} 0{,}22 \\ 3{,}12 \\ 1{,}57 \\ 3{,}12 \\ 2{,}20 \\ 0{,}22 \end{pmatrix}, \bar{x} = 1{,}74$$

Bagging

- Bagging = **B**ootstrap **agg**regat**ing**
- Bagging – Algorithmus:
  - Wiederhole M mal:
    - Bestimme Bootstrap-Sample L_k von L.
    - Bestimme Vorhersagemodell für L_k

z.B. Modellbaum

- Kombiniere die M gelernten Vorhersagemodelle:
  - Klassifikationsprobleme: Voting
  - Regressionsprobleme: Mittelwertbildung

- Originalreferenz:
  L.Breiman: „Bagging Predictors". 1996

Random Forests

Bootstrap

- Wiederhole:
  - Ziehe zufällig mit Zurücklegen n Beispiele aus der Trainingsmenge
  - Wähle zufällig $m' < m$ Merkmale
  - Lerne Entscheidungsbaum (ohne Pruning)

- Klassifikation: Maximum über alle Bäume (Voting)
- Regression: Mittelwert über alle Bäume

- Originalreferenz:

  L. Breiman: „Random Forests". 2001

Zusammenfassung Entscheidungsbäume

- Klassifikation : Vorhersage diskreter Werte.
  - Diskrete Attribute: ID3.
  - Kontinuierliche Attribute: C4.5.
  - Skalierbar für große Datenbanken: SLIQ, SPRINT.
- Regression: Vorhersage kontinuierlicher Werte.
  - Regressionsbäume: konstante Werte in Blättern.
  - Modellbäume: lineare Modelle in den Blättern.