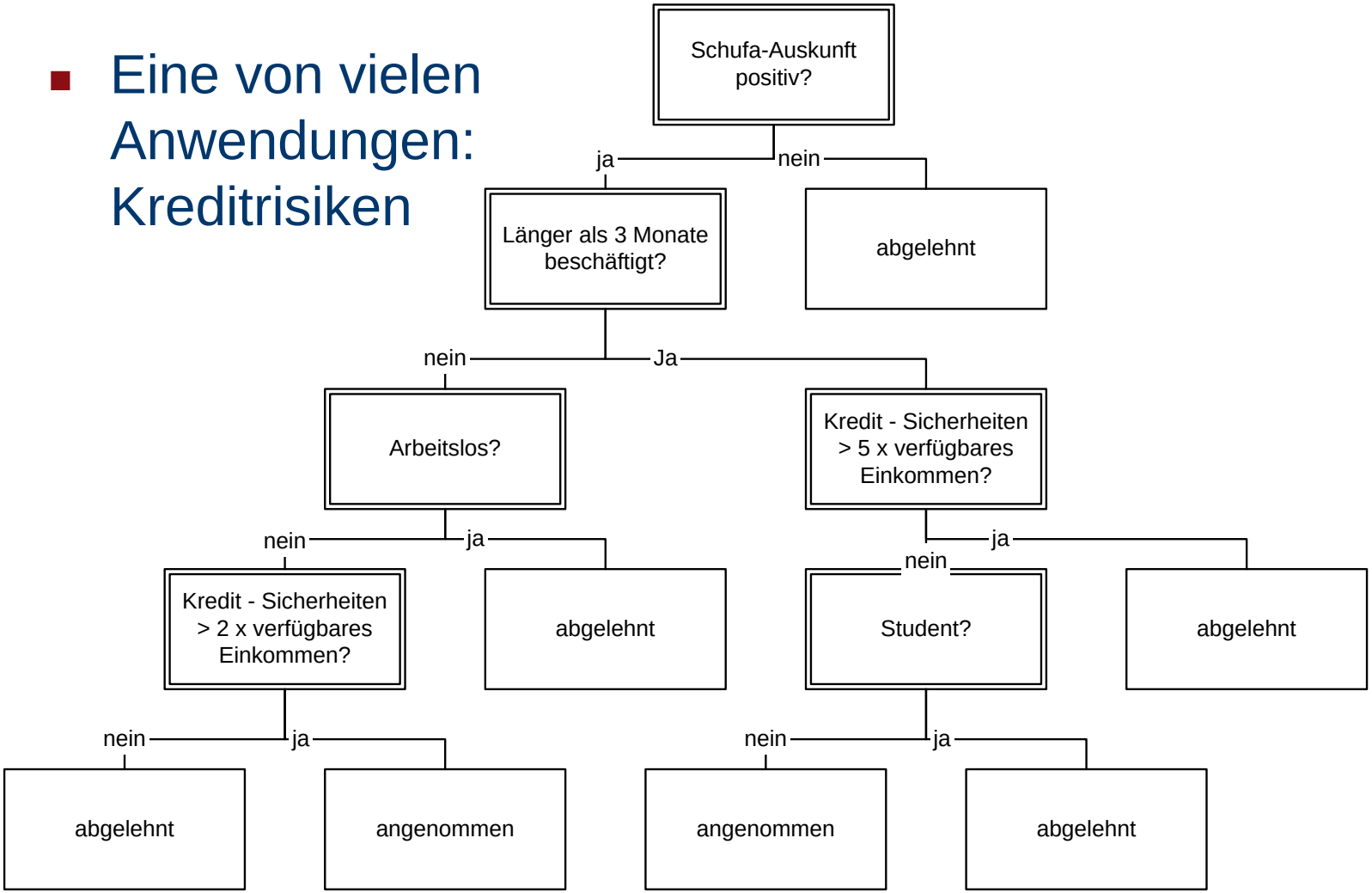
Universität Potsdam
Institut für Informatik
Lehrstuhl Maschinelles Lernen

Entscheidungsbäume

Christoph Sawade/Niels Landwehr
Jules Rasetaharison,
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.
  - $L = \langle (\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N) \rangle$
  - $\mathbf{x}_i = \begin{pmatrix} x_{i1} \\ \vdots \\ x_{im} \end{pmatrix}$
- Ausgabe: Klassifikator.
  - $f : X \to Y$

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.

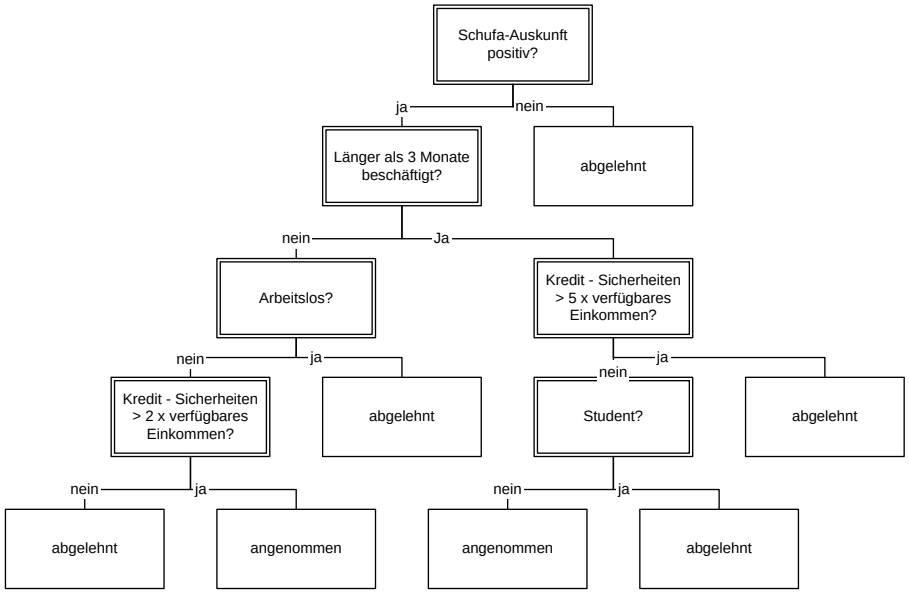
- Bei Lernen: kontinuierliche Trainingsdaten.
  - $L = \langle (\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N) \rangle$
  - z.B. $\langle (\mathbf{x}_1, 3.5), ..., (\mathbf{x}_N, -2.8) \rangle$

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

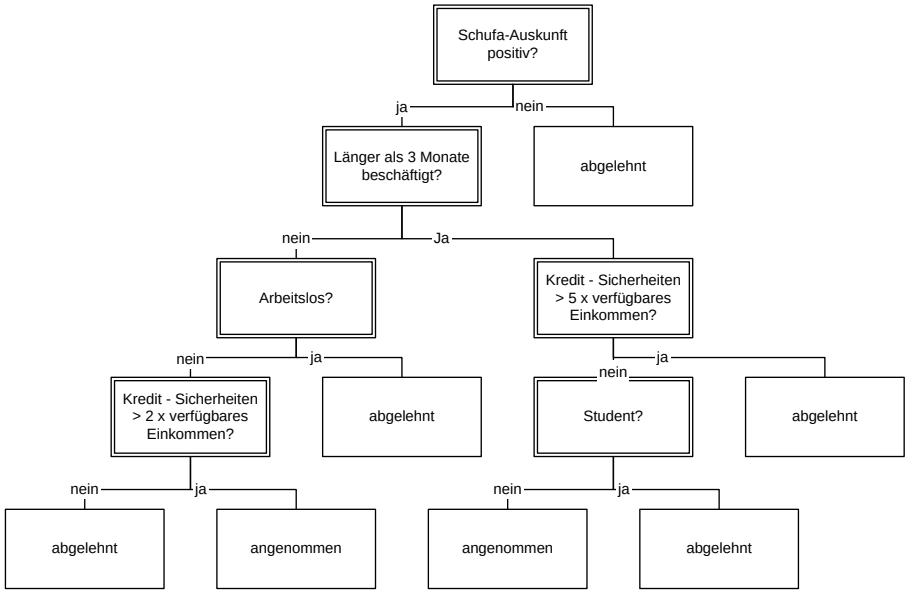
- Testknoten: Testen, ob der Wert eines Attributs eine Bedingung erfüllen; bedingte Verzweigung in einen Ast.
- Terminalknoten: Liefern einen Wert als Ausgabe.



Anwendung von Entscheidungsbäumen

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

- Terminalknoten: liefere Wert als Klasse zurück.



Lernen von Entscheidungsbäumen

- Finde Entscheidungsbaum, der zumindest für die Trainingsdaten die richtige Klasse liefert.
- 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?

Algorithmus ID3

- Voraussetzung:
  - Klassifikationslernen,
  - Alle Attribute haben festen, diskreten Wertebereich.
- Idee: rekursiver Algorithmus.
  - Wähle das Attribut, der die Unsicherheit bzgl. der Klasse 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: $L = \langle (\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N) \rangle$, verfügbar= $(x_1, ..., x_m)$
- 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 , $\text{verfügbar} \setminus x_*$).

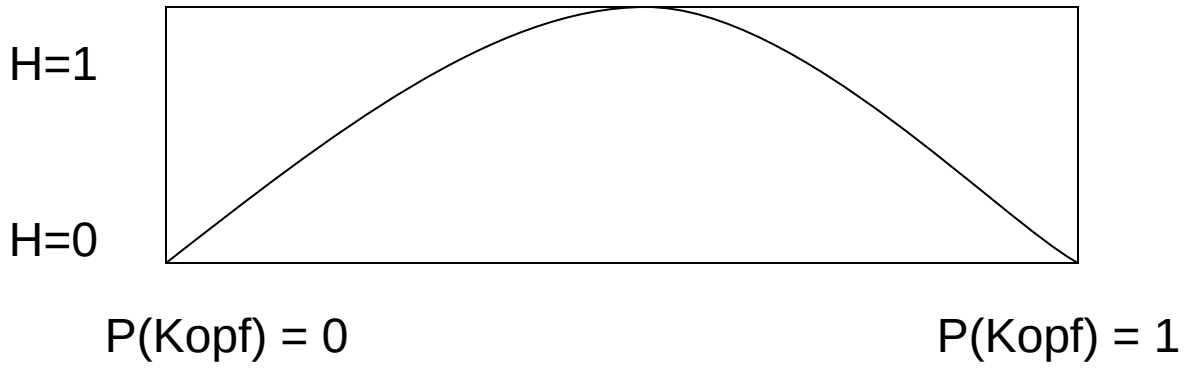
Algorithmus ID3: Beispiel

Beispiele	x_1: Kredit > 3 x Einkommen	x_2: Länger als 3 Monate beschäftigt	x_ 3: Student?	y: Kredit zurückgezahlt?
x1	+	+	-	-
x2	+	-	+	-
x3	-	+	+	+
x4	-	-	+	+
x5	-	+	-	-
x6	-	+	-	-

- Eingabe: $L = \langle (\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N) \rangle$, verfügbar= $(x_1, ..., x_m)$
- 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 , $\text{verfügbar} \setminus x_*$).

Information und Entropie

- Information = Maß für Ungewissheit = Entropie.
- Entropie einer Zufallsvariable y:
  - $H(y) = -\sum_v p(y=v) \log_2 p(y=v)$
- Empirische Entropie Zufallsvariable y auf Daten L:
  - $H(L,y) = -\sum_v \hat{p}_L(y=v) \log_2 \hat{p}_L(y=v)$
- Stetig erweitert bei p(v)=0
- Entropie eines Münzwurfs:



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:
  - $$H_{|x=v}(L,y) = H(L,y \mid x=v)$$
    $$= -\sum_{v'} \hat{p}_L(y=v' \mid x=v) \log_2 \hat{p}_L(y=v' \mid x=v)$$
- Stetig erweitert bei p=0.

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

≈0.25 ≈0.04 ≈0.46

Beispiele	x_1: Kredit > 3 x Einkommen	x_2: Länger als 3 Monate beschäftigt	*x*_ 3: Student?	y: Kredit zurückgezahlt?
x1	+	+	-	-
x2	+	-	+	-
x3	-	+	+	+
x4	-	-	+	+
x5	-	+	-	-
x6	-	+	-	-

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

≈0.92

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

Information Gain / Info Gain Ratio

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

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.
  - $$GainRatio(L,x) = \frac{IG(L,x)}{SplitInfo(L,x)}$$
  - $$SplitInfo(L,x) = -\sum_v \frac{|L_{|x=v}|}{|L|} \log_2 \frac{|L_{|x=v}|}{|L|}$$

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.

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: $L = \langle (\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N) \rangle$.
- 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: $[x_* \le v_*] = \arg\max_{\text{Attribute } x_i, \text{ Werte } v \text{ in } L} IG(L, [x_i \le v])$
  - Wenn bestes Attribut diskret, Rekursion wie ID3.
  - Wenn bestes Attribut kontinuierlich, teile Trainingsmenge:
    - $L_{links} = \langle (\mathbf{x}_k, y_k) \in L \mid x_{k*} \le v_* \rangle$, $L_{rechts} = \langle (\mathbf{x}_k, y_k) \in L \mid x_{k*} > v_* \rangle$
    - 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:
  - $IG(L,[x \leq v]) = H(L,y) - \hat{p}_L([x \leq v])H(L_{\llbracket x \leq v \rrbracket}, y) - \hat{p}_L([x > v])H(L_{\llbracket x > v \rrbracket}, y)$

C4.5: Beispiel

Pruning

- Problem: Blattknoten, die nur von einem (oder sehr wenigen) Beispielen gestützt werden, sind 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 müssen.

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

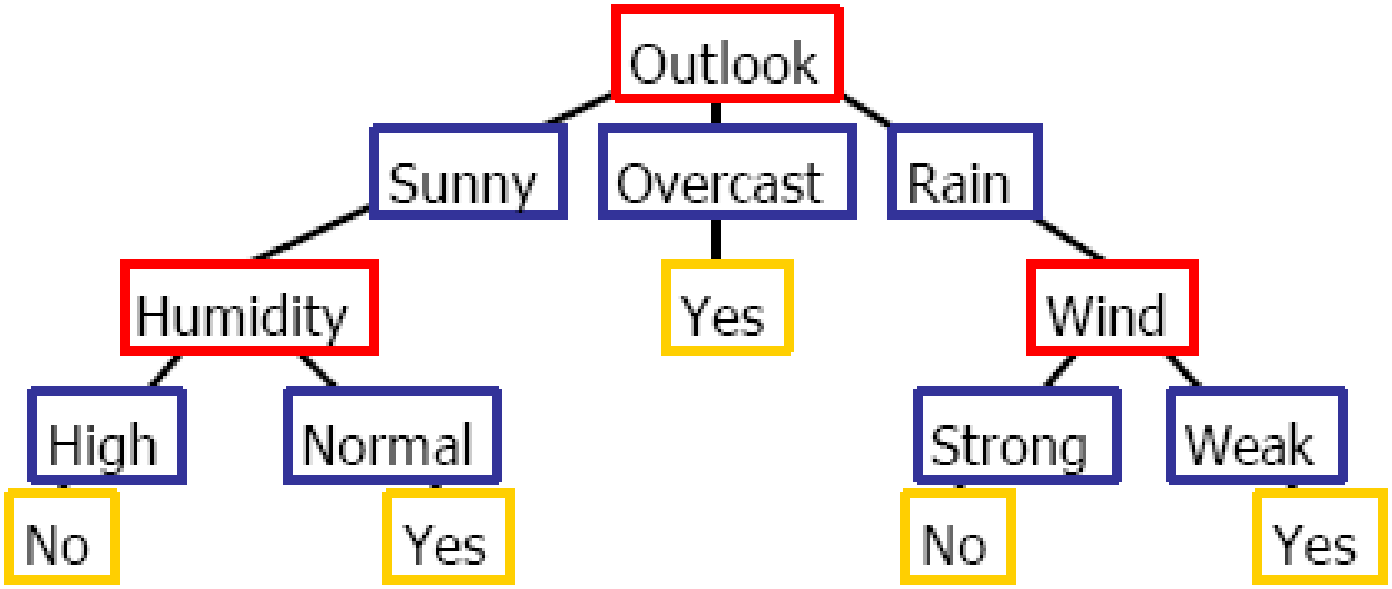
- Aufteilen der Trainingsdaten in Trainingsmenge und Pruningmenge.
- Nach dem Aufbau des Baumes mit der Trainingsmenge:
  - Versuche, zwei Blattknoten durch Löschen eines Tests zusammenzulegen,
  - Solange dadurch die Fehlerrate auf der Pruningmenge verringert wird.

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: If (Outlook=Sunny) ? (Humidity=High) Then PlayTennis=No
R_2: If (Outlook=Sunny) ? (Humidity=Normal) Then PlayTennis=Ye
R_3: If (Outlook=Overcast) Then PlayTennis=Yes
R_4: If (Outlook=Rain) ? (Wind=Strong) Then PlayTennis=No
R_5: If (Outlook=Rain) ? (Wind=Weak) Then PlayTennis=Yes

Induktiver Bias

- 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

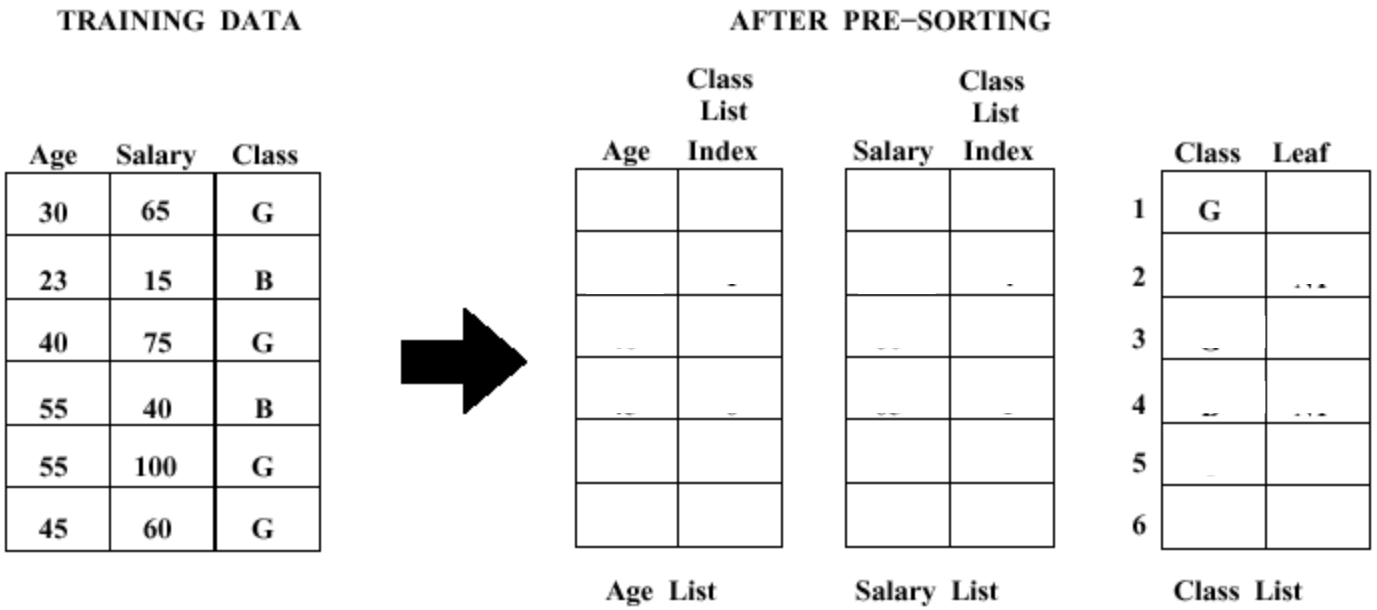
1. Starte **Vorsortieren** der Beispiele
2. Solange Abbruchkriterium noch nicht erreicht ist
    1. Für alle Attribute
        1. Lege für alle Knoten ein Klassenhistogramm an.
        2. Starte **Evaluierung** der Splits
    2. Wähle Split
    3. Aktualisiere Entscheidungsbaum; für jeden neuen Knoten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

SLIQ (1. Teilschritt)

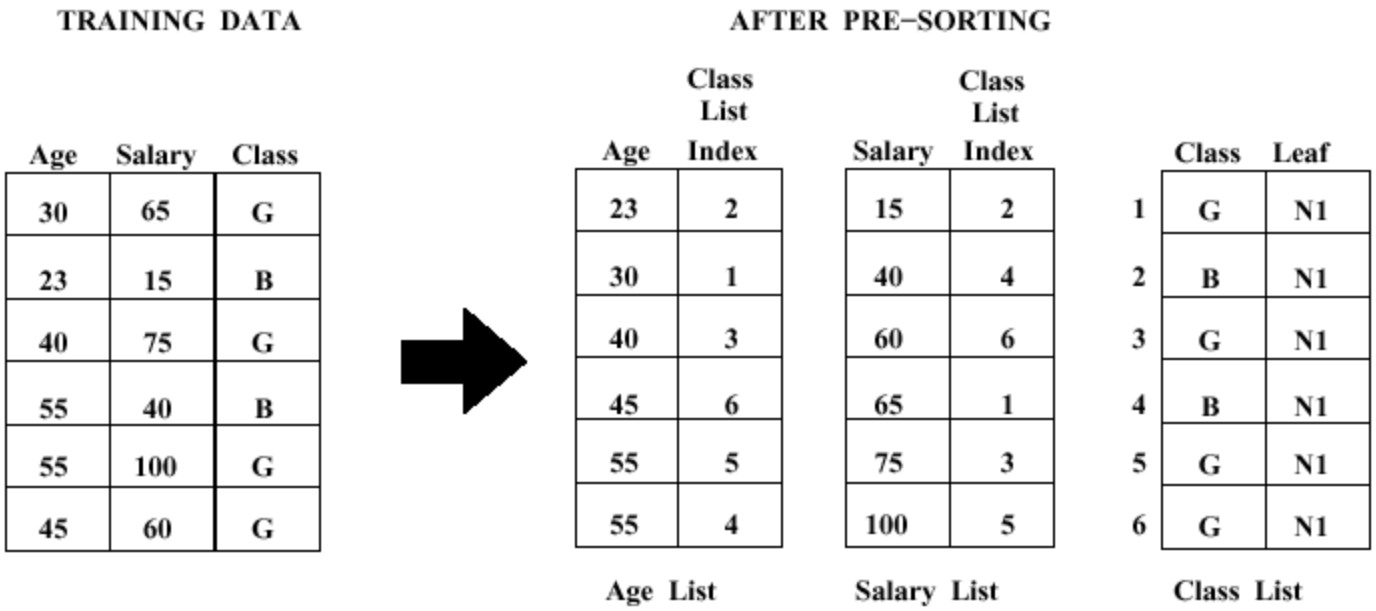
- Vorsortieren der Beispiele

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



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 (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

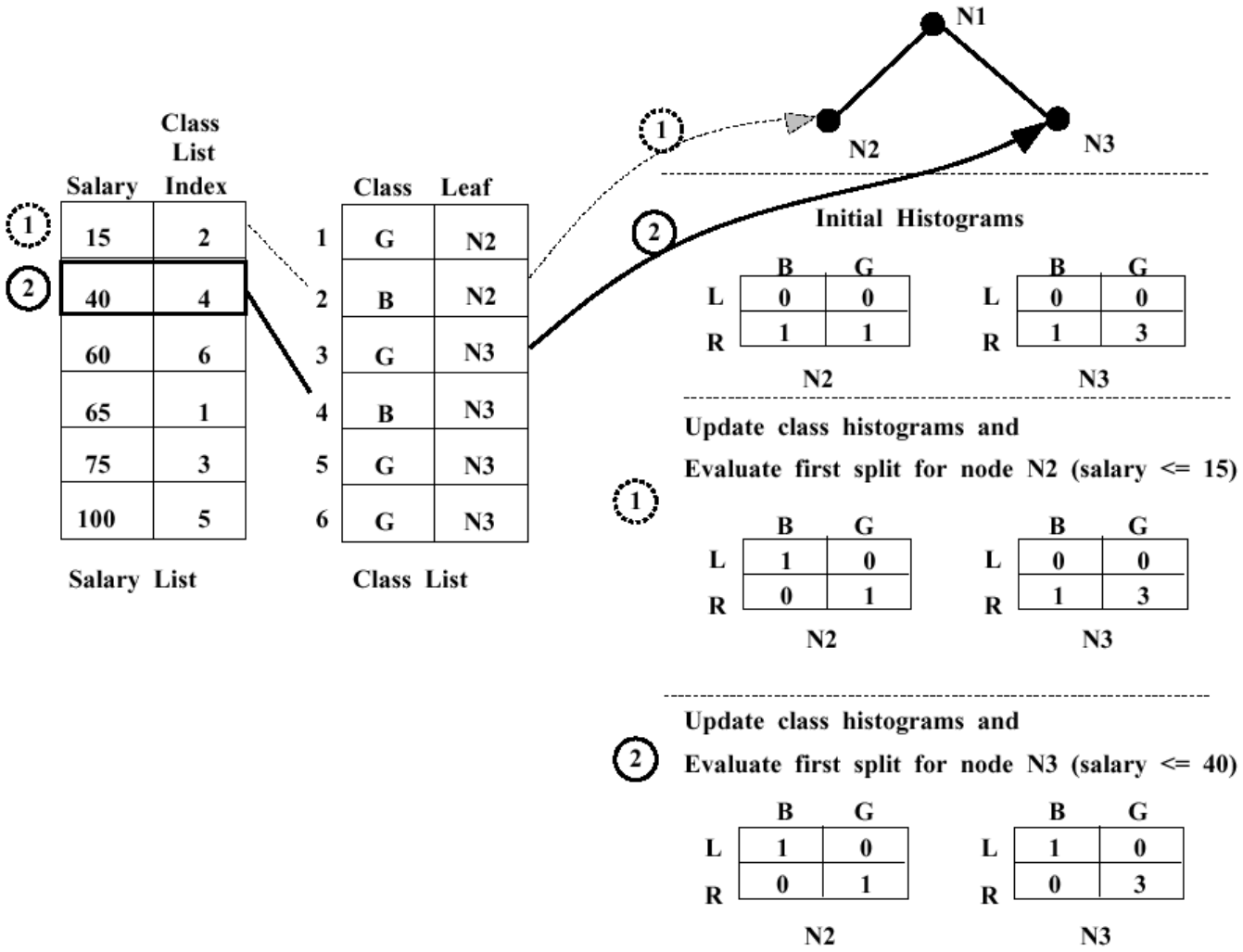
1. Starte **Vorsortieren** der Beispiele ✓
2. Solange Abbruchkriterium noch nicht erreicht ist
    1. Für alle Attribute
        1. Lege für alle Knoten ein Klassenhistogramm an.
        2. Starte **Evaluierung** der Splits ✓
    2. Wähle Split
    3. Aktualisiere Entscheidungsbaum; für jeden neuen Knoten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

SLIQ (3. Teilschritt)

- Aktualisiere Klassenliste (Knoten)

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

SLIQ: Beispiel



SLIQ: Datenstrukturen

- Datenstrukturen im Speicher
- Ausgelagerte Datenstrukturen
- Datenstrukturen in Datenbank

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

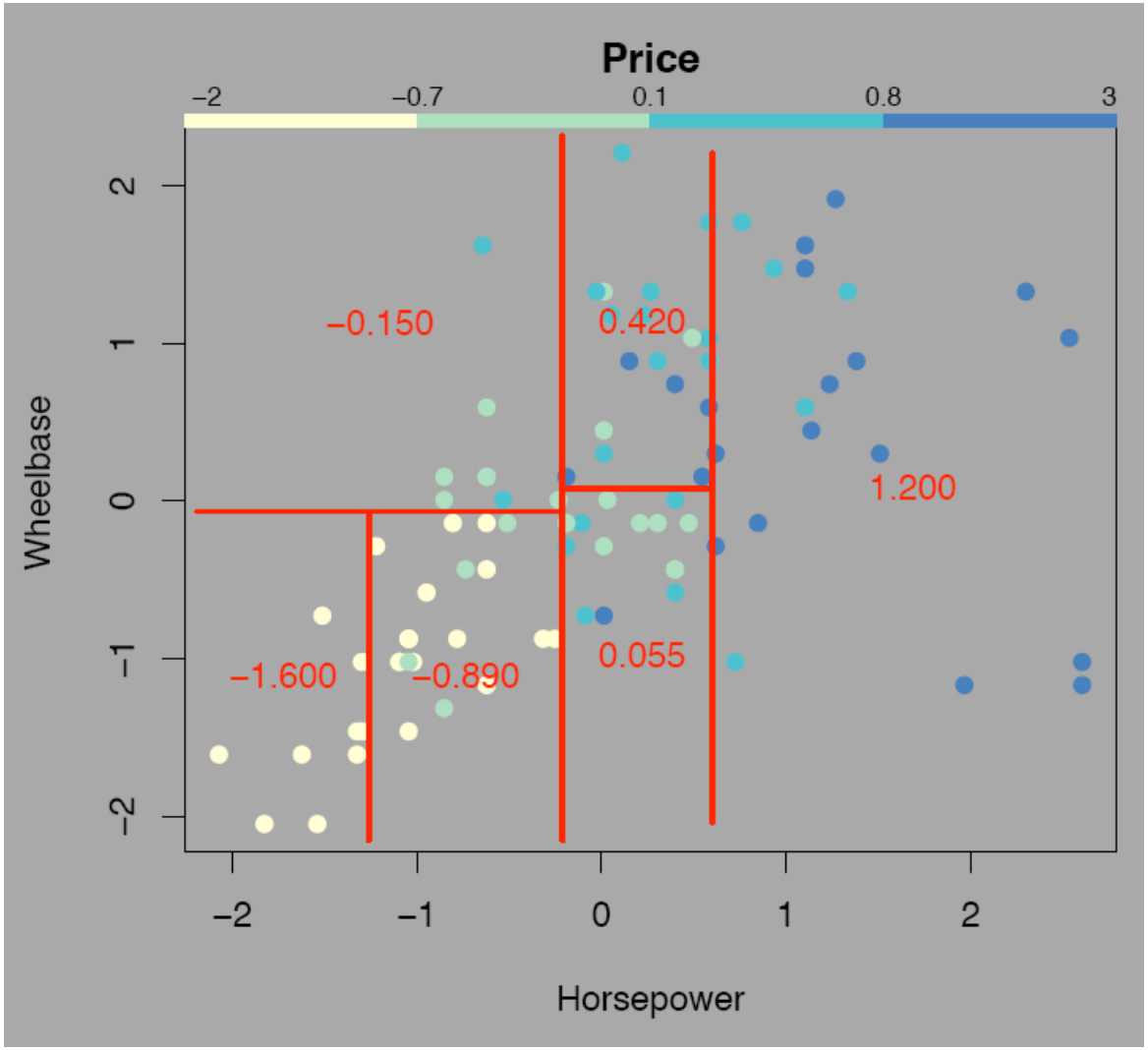
- Eingabe: $L = \langle (\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N) \rangle$, kontinuierliche y.
- Gesucht: $f : X \rightarrow Y$
- Algorithmus CART
  - Wurde zur gleichzeitig, unabhängig von C4.5 entwickelt.
  - Terminalknoten enthalten kontinuierlichen Wert.
  - Algorithmus wie C4.5. Für Klassifikation 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: $SSE_L = \sum_{(x,y)\in L} (y - \bar{y})^2$ mit $\bar{y} = \frac{1}{|L|} \sum_{(x,y)\in L} y$
- Mit welchem Test soll gesplittet werden?
- Kriterium für Testknoten $[x \leq v]$:
  - $\text{SSE - Red}(L, [x \leq v]) = SSE_L - SSE_{L|[x\leq v]} - SSE_{L|[x>v]}$
  - 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.

Regressionsbäume

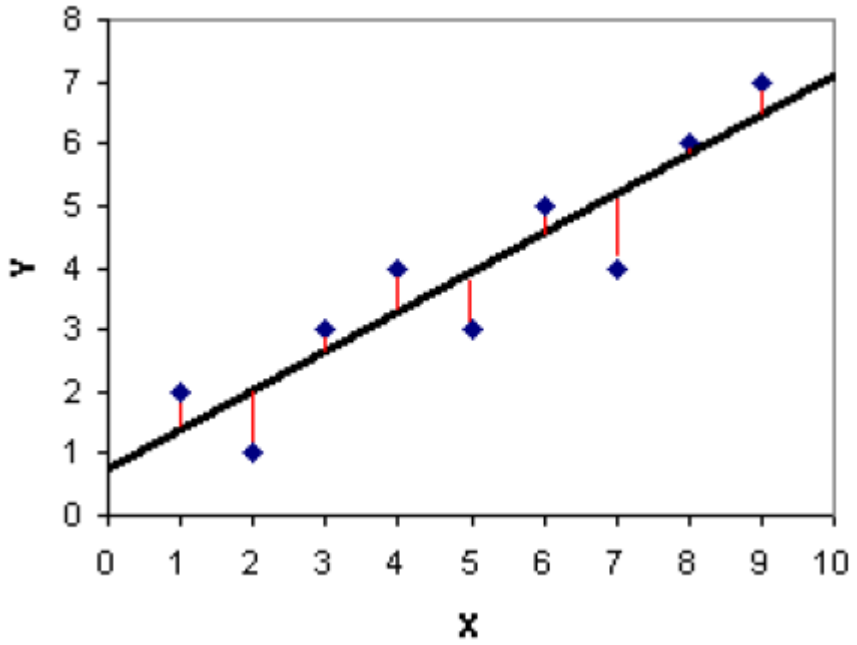


Lineare Regression

- 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

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

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

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

Lineare Regression, eindimensional

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

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

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

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

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

Lineare Regression

- Eingabe: $L = \langle (\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N) \rangle$
- Gesucht: lineares Modell $f(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\mathbf{x} + c$
- kleiner Trick:
  - $f(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\mathbf{x} + c$

$$= (w_1 \quad \ldots \quad w_m) \begin{pmatrix} x_1 \\ \ldots \\ x_m \end{pmatrix} + c$$

$$= (w_1 \quad \ldots \quad w_m \quad c) \begin{pmatrix} x_1 \\ \ldots \\ x_m \\ 1 \end{pmatrix} = \mathbf{w'}^{\mathrm{T}}\,\mathbf{x'}$$

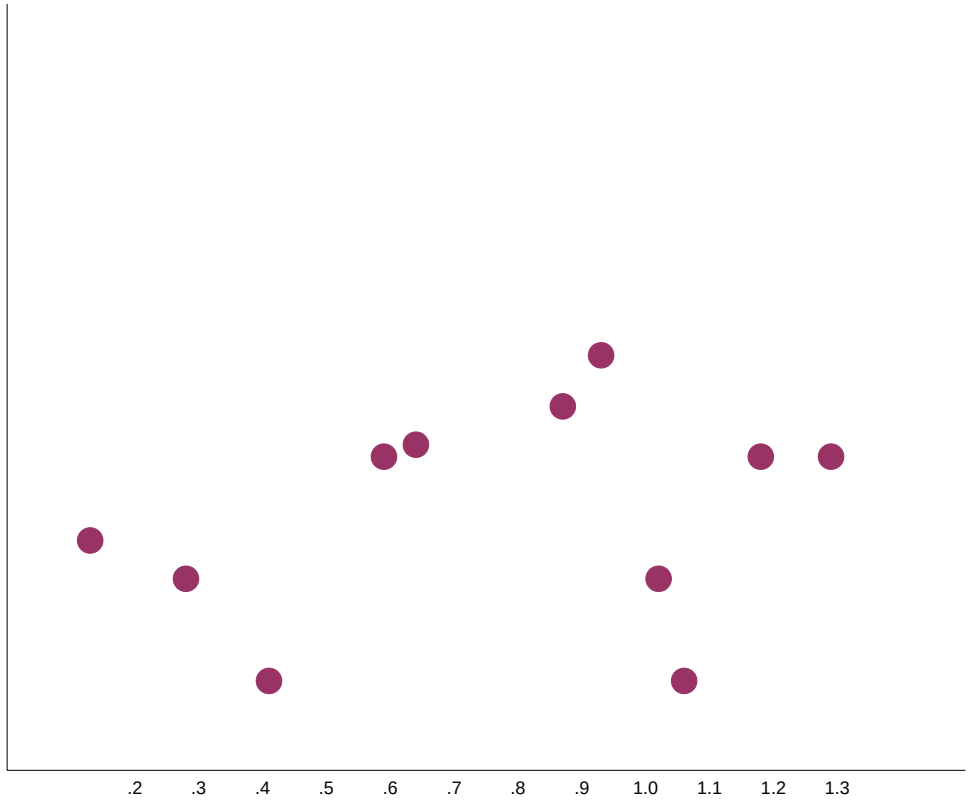
- c kann weg, dafür eine zusätzliche Dimension.

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 (siehe Ridge Regression).

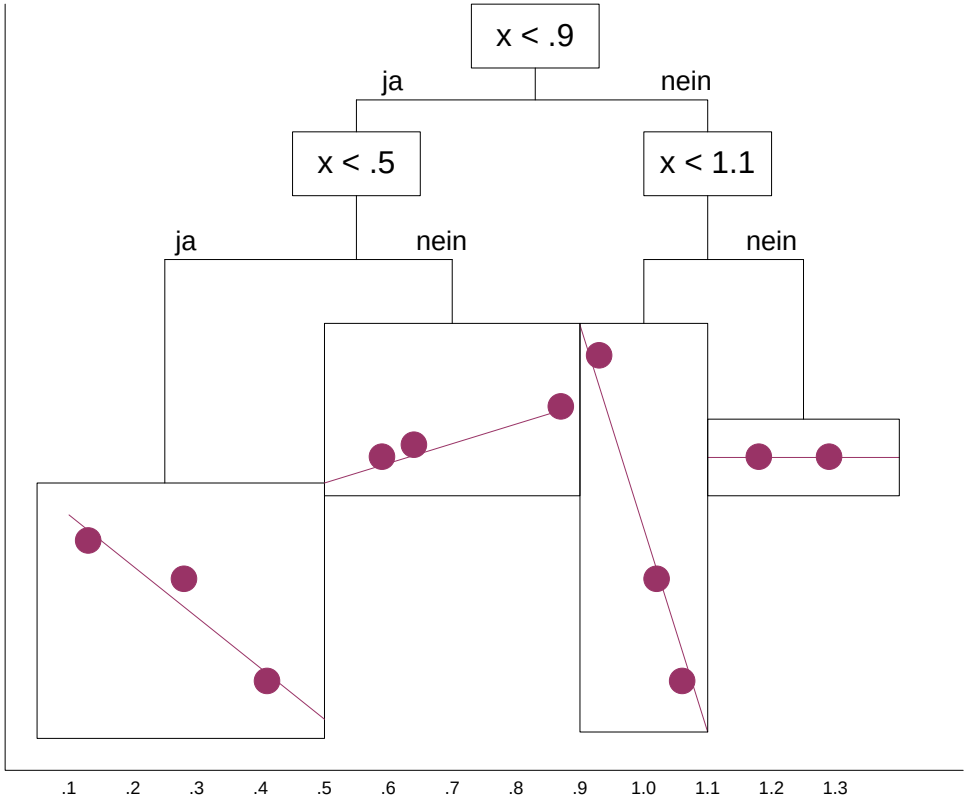
Modellbäume

- 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.

Entscheidungsbaum - Vorteile

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

Entscheidungsbaum - Nachteile

Entscheidungsbaum - Nachteile

- Nicht robust gegenüber Rauschen
- Tendenz zum Overfitting
- Instabil

Bagging / Bootstrapping

- Ziel: Varianzminimierung
- Bootstrapping
  - Wiederholtes gleichverteiltes Ziehen mit Zurücklegen von Beispielen aus einem Sample.
- Bagging (**B**ootstrap **agg**regat**ing)**
  - Lernen eines Ensembles von Modellen aus Bootstrap-Samples.
  - Vgl. Model Averaging
- Originalreferenz:
  L.Breiman: „Bagging Predictors”. 1996

Random Forests

- Wiederhole:
  - Ziehe zufällig mit Zurücklegen n Beispiele aus der Trainingsmenge
  - Wähle zufällig $m' \ll m$ Merkmale
  - Lerne Entscheidungsbaum (ohne Pruning)
- Klassifikation: Maximum über alle Bäume
- Originalreferenz:

  L. Breiman: „Random Forests”. 2001

Bootstrap

Entscheidungsbäume

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