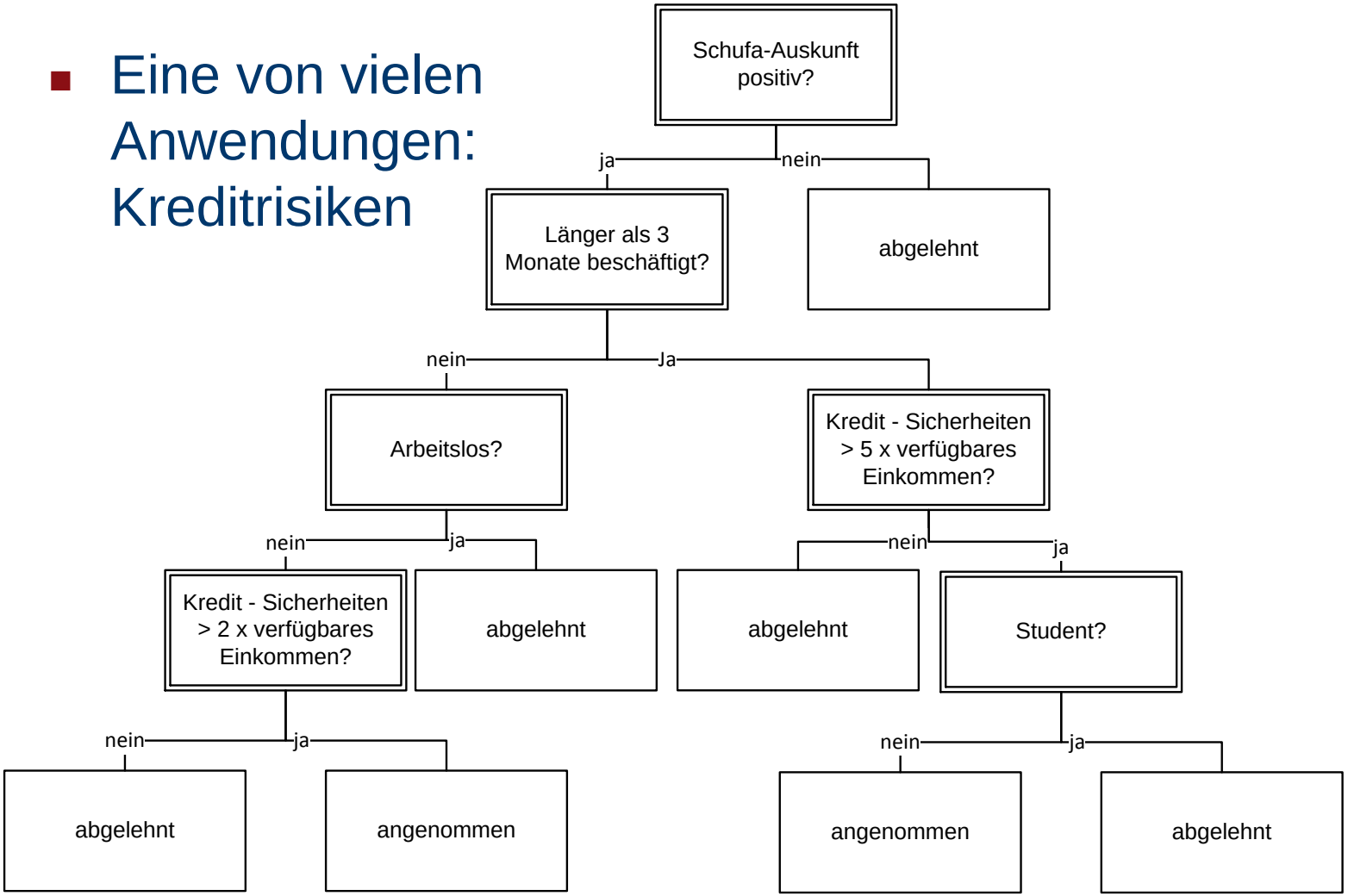
Universität Potsdam
Institut für Informatik
Lehrstuhl Maschinelles Lernen

Maschinelles Lernen
Entscheidungsbäume

Paul Prasse

Entscheidungsbäume

- Eine von vielen Anwendungen: Kreditrisiken



Entscheidungsbäume – Warum?

- 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.
- Verschiedene Lernprobleme werden abgedeckt
  - 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.
  - $\mathbf{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_m \end{pmatrix}$ z.B. Merkmale wie Farbe, Testergebnisse, Form, …
- Ausgabe: Klasse $y \in Y$; endliche Menge Y.
  - Beispiele:
    - {akzeptiert, abgelehnt};
    - {Kind, Frau, Mann}.
  - Klasse wird auch als Zielattribut bezeichnet.

Klassifikationslernen

- Eingabe: Trainingsdaten.
  - $L = \langle (\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n) \rangle$
  - $\mathbf{x}_i = \begin{pmatrix} x_1 \\ \vdots \\ x_m \end{pmatrix}$
- Ausgabe: Klassifikator.
  - $f : X \to Y$

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.

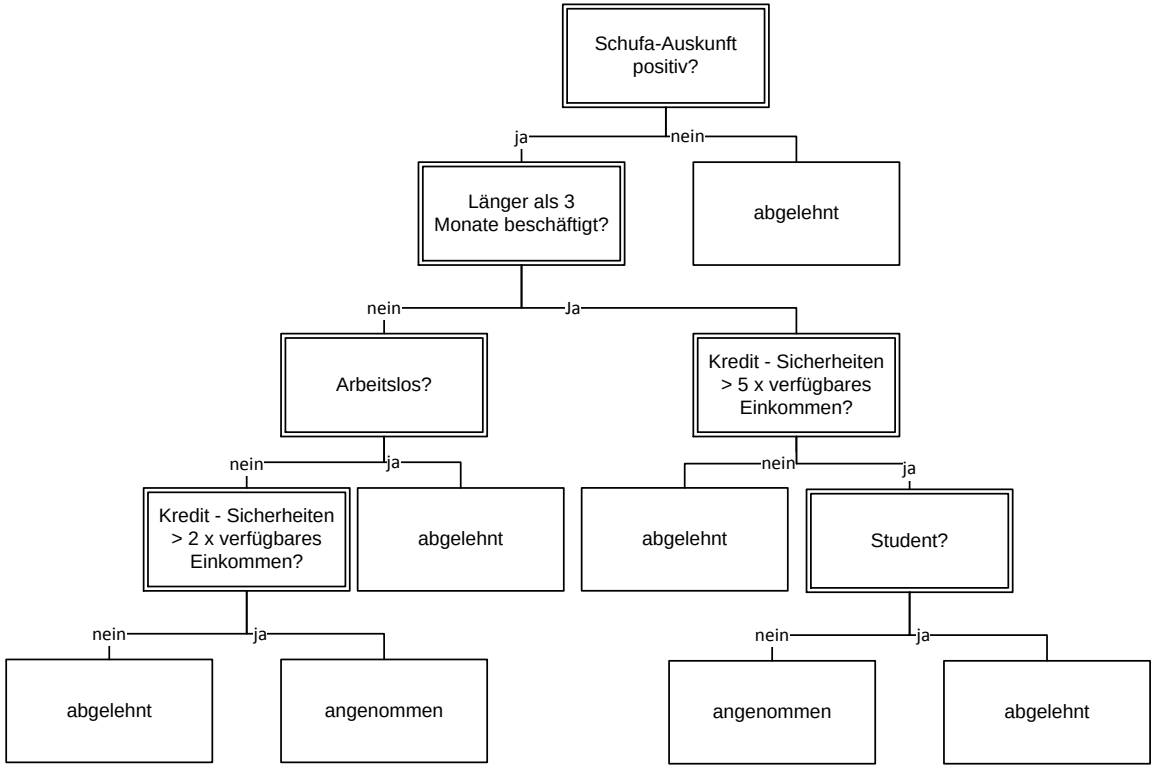
- Lernproblem: kontinuierliche Trainingsdaten.
  - $L = \langle (\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n) \rangle$
  - z.B. $\langle (\mathbf{x}_1, 3.5), ..., (\mathbf{x}_N, -2.8) \rangle$
  - Temperatur, Körpergröße, Bindungsstärke

Andere Lernprobleme

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

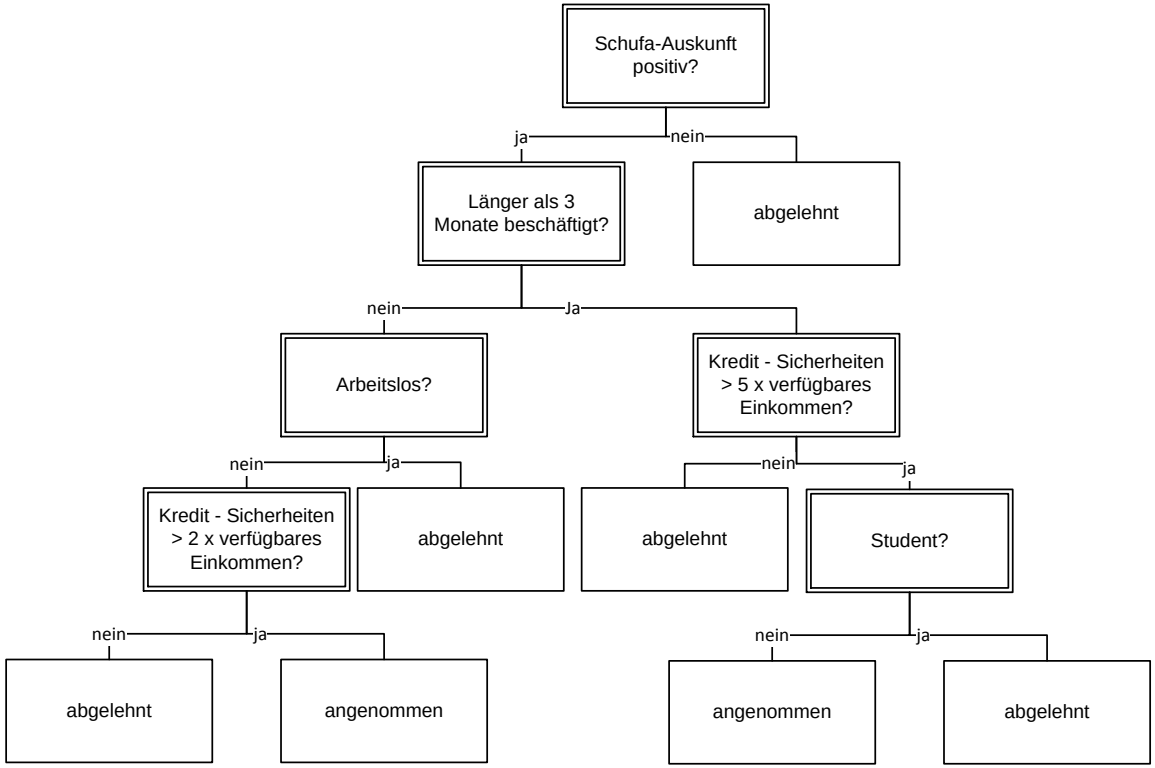
Entscheidungsbäume - Aufbau

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



Entscheidungsbäume - Repräsentation

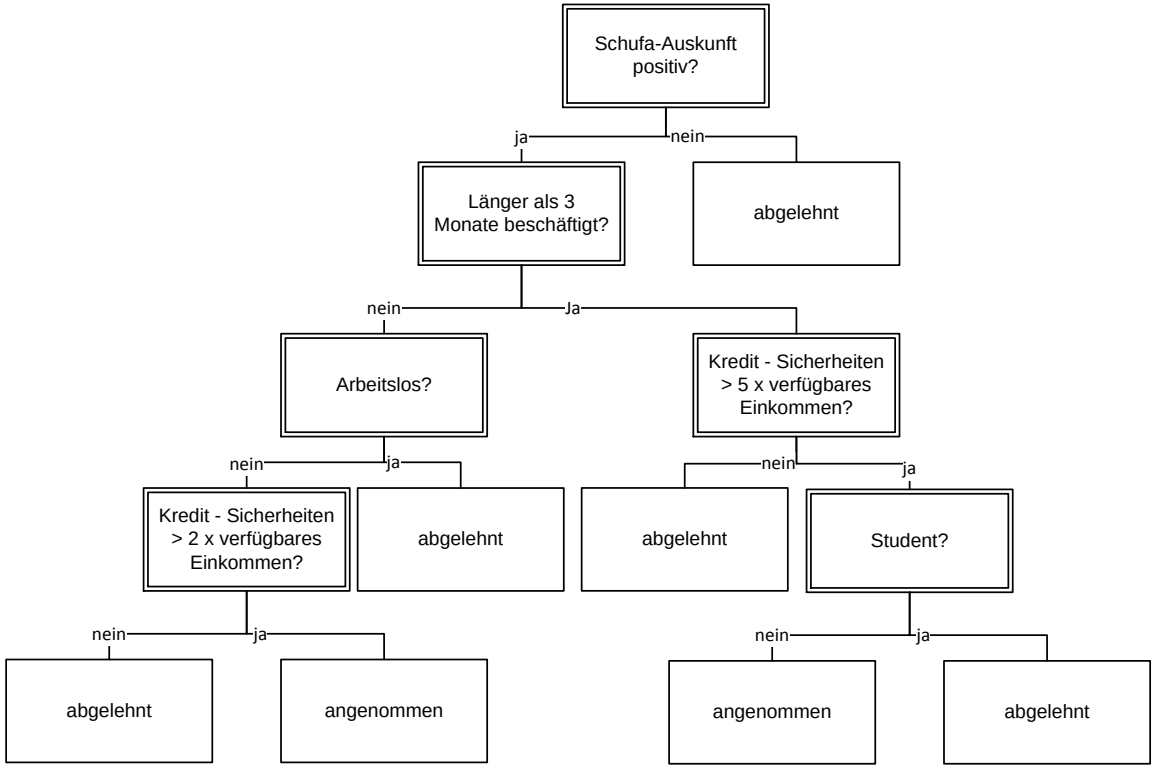
- Entscheidungsbäume können logische Operationen repräsentieren, wie:
  - $\wedge, \vee, XOR$
  - $(A \wedge B) \vee (C \wedge \neg D \wedge E)$



Anwendung von Entscheidungsbäumen

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

- Terminalknoten: Liefere Wert als Klasse zurück.



Entscheidungsbäume – Wann sinnvoll?

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

- Anwendungsgebiete:
  - Medizinische Diagnose
  - Kreditwürdigkeit
  - Vorverarbeitungsschritt in der Computer Vision (z.B. Objekterkennung)

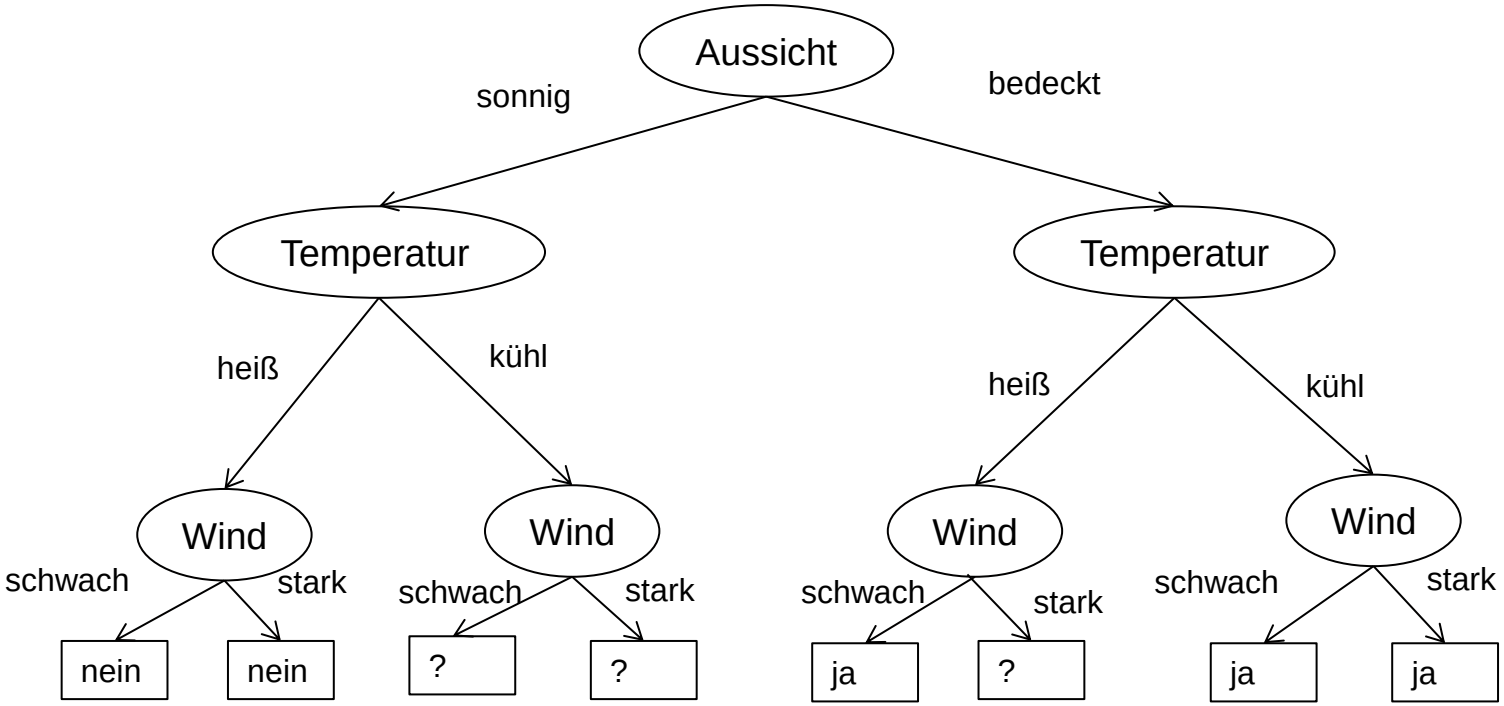
Lernen von Entscheidungsbäumen

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 repräsentiert.

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.
    - 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.

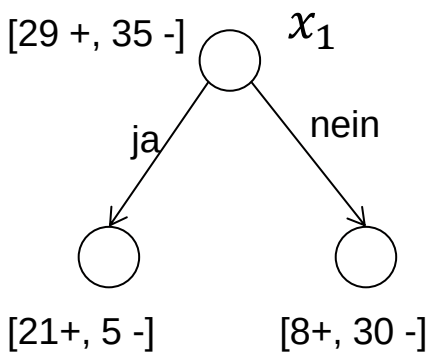
- Komplexität einer vollständigen Suche nach kleinstem Baum?

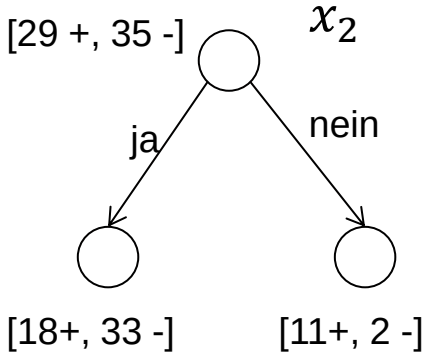
Lernen von Entscheidungsbäumen

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

Greedy-Algorithmus – Top-Down Konstruktion

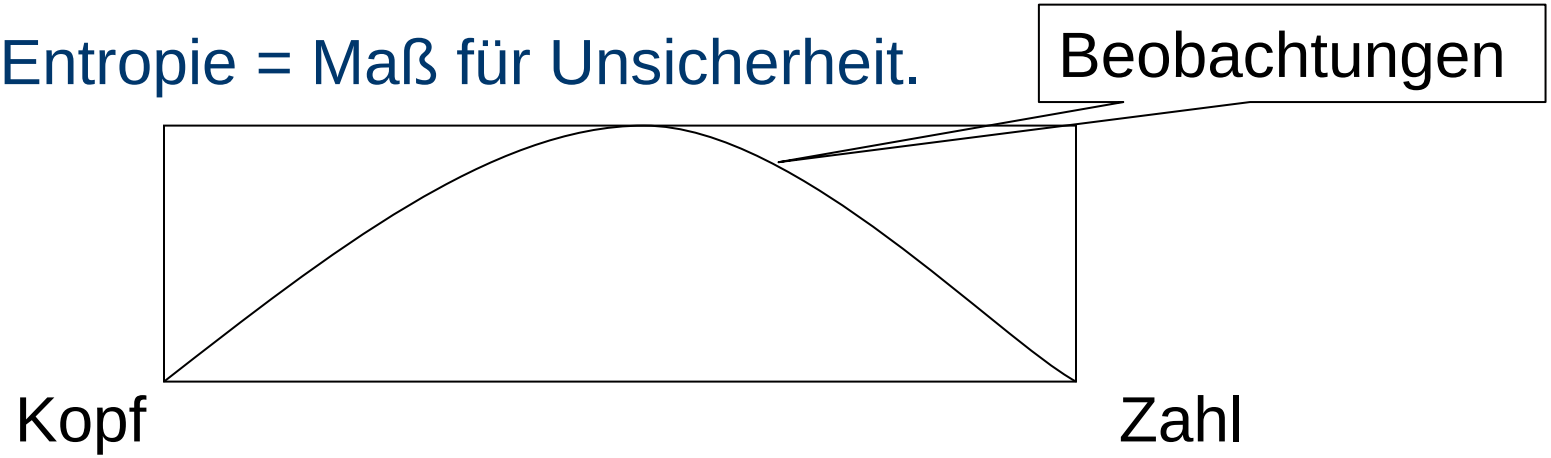
- Solange die Trainingsdaten nicht perfekt klassifiziert sind:
  - Wähle das „beste“ Attribut A, um die Trainingsdaten zu trennen.
  - Erzeuge für jeden Wert des Attributs A einen neuen Nachfolgeknoten und weise die Trainingsdaten dem so entstandenen Baum zu.
- Problem: Welches Attribut ist das beste?





Splitkriterium: 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 :
  - $H(S) = -p_+ \log_2 p_+ \; - p_- \log_2 p_-$

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

Splitkriterium: 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:
  - $H_{|x=v}(L,y) = H(L,y \mid x=v)$

$$= -\sum_{v'} \hat{p}_L(y=v' \mid x=v) \log_2 \hat{p}_L(y=v' \mid x=v)$$

Splitkriterium: Information Gain

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

Beispiel – Information Gain

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 – Information Gain

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

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

Information Gain / Gain Ratio

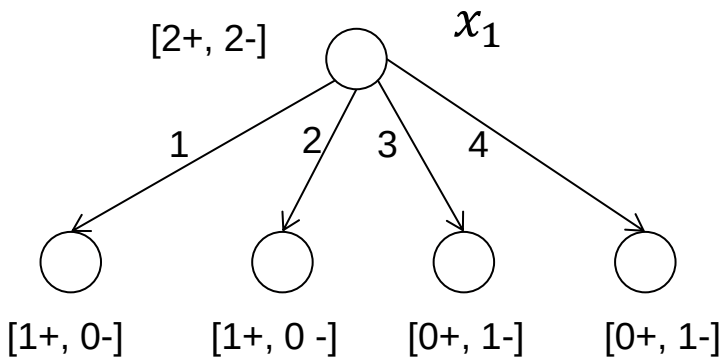
- Motivation:
  - Vorhersage ob ein Student die Prüfung besteht.
  - Wie hoch ist der Information Gain des Attributes „MatrikelNummer“?
  - Informationsgehalt des Tests ist riesig.

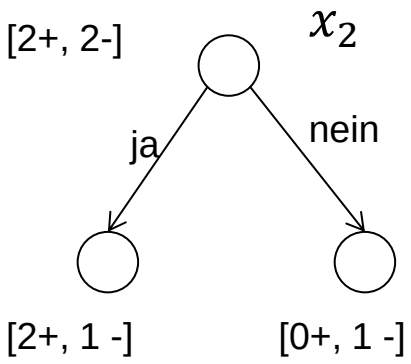
Information Gain / Gain Ratio

- Motivation:
  - Vorhersage ob ein Student die Prüfung besteht.
  - Wie hoch ist der Information 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
    - Information Gain, Gain Ratio, Gini Index
  - 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: $L = \langle (\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n) \rangle$, verfügbare Attribute = $(x_1, ..., x_m)$

- Wenn alle Beispiele in L dieselbe Klasse y haben,
  - dann gib Terminalknoten mit Klasse y zurück.
- Wenn Menge der verfügbaren Attribute =∅,
  - 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 , 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: $L = \langle (\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n) \rangle$, verfügbare Attribute = $(x_1, ..., x_m)$

- Wenn alle Beispiele in L dieselbe Klasse y haben,
  - dann gib Terminalknoten mit Klasse y zurück.
- Wenn Menge der verfügbaren Attribute =ø,
  - 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 , verfügbar $\setminus x_*$).

Kontinuierliche Attribute

- ID3 wählt Attribute mit größtem Informationsgehalt aus und bildet dann einen Zweig für jeden Wert dieses Attributes.
- Problem: 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

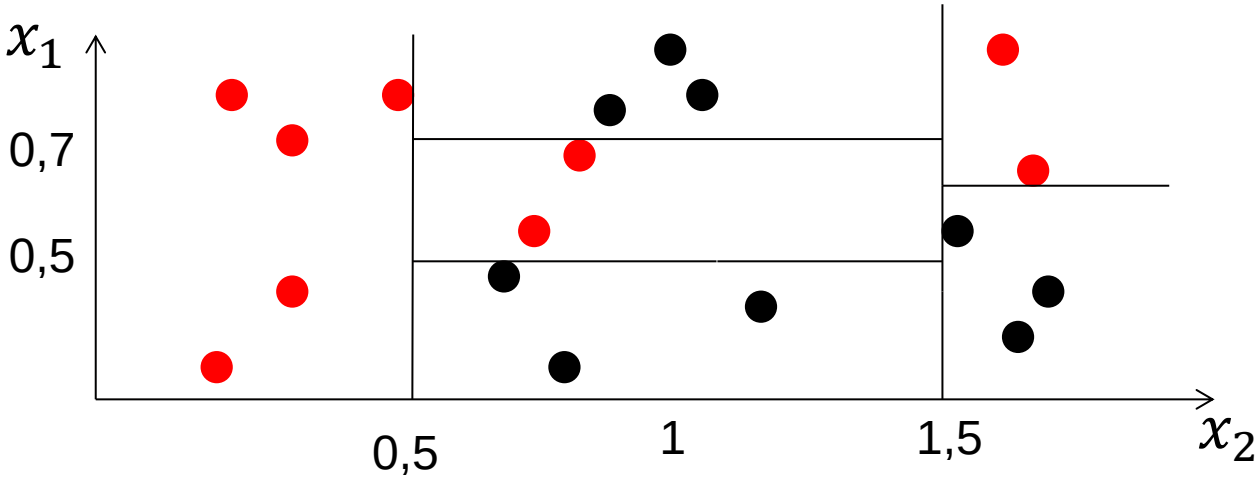
- Eingabe: $L = \langle (\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n) \rangle$.

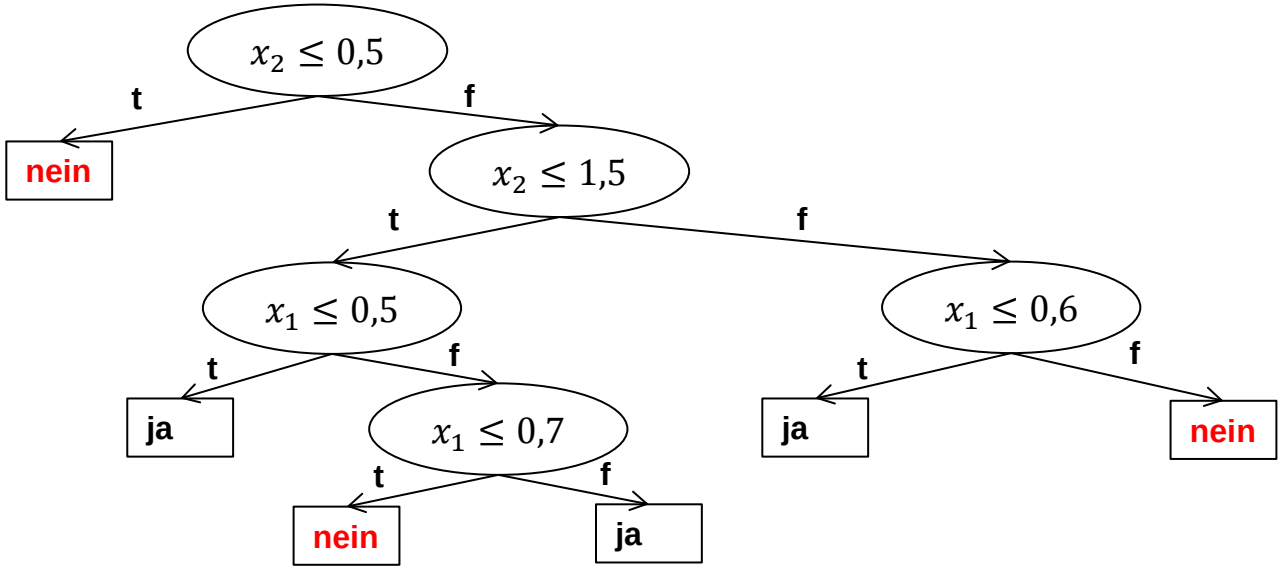
- Wenn alle Beispiele in L dieselbe Klasse y haben,
  - dann gib Terminalknoten mit Klasse y zurück.
- Wenn alle Instanzen identisch sind:
  - Gib Terminalknoten mit häufigster Klasse in L zurück.
- Sonst konstruiere besten Testknoten, iteriere dazu über alle Attribute.
  - Diskrete Attribute: wie ID3.
  - 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:
  - $IG(L,[x \leq v]) = H(L,y) - \hat{p}_L([x \leq v])H(L_{\|[x \leq v]}, y) - \hat{p}_L([x > v])H(L_{\|[x > v]}, y)$

C4.5: Beispiel





Pruning

- Problem: Blattknoten, die nur von einem (oder sehr wenigen) Beispielen gestützt werden, liefern häufig keine gute Klassifikation(Generalisierungsproblem).
- Pruning: Entferne 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 vorraus.

- Regularisierungsparameter r.
- Einstellung mit Cross Validation.

Reduced Error Pruning

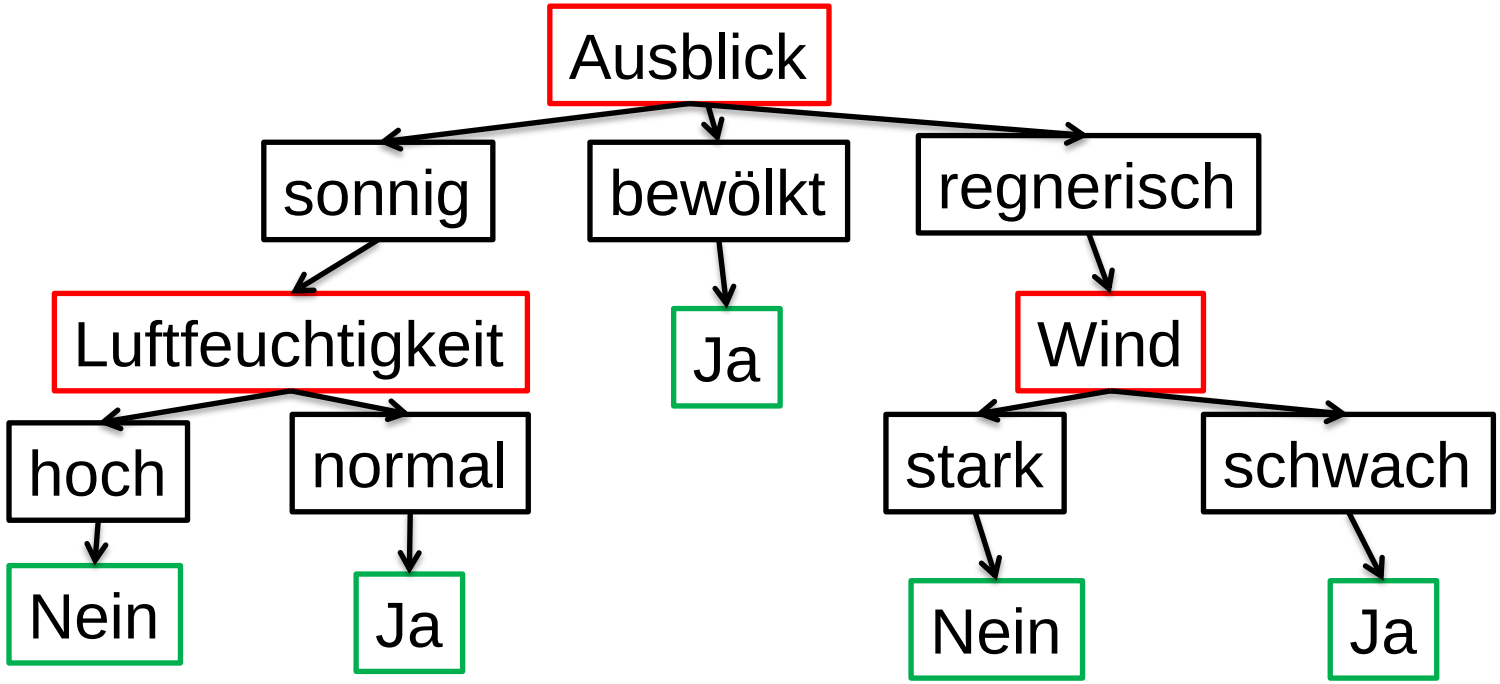
- Aufteilen der Trainingsdaten in Trainingsmenge und Pruningmenge.
- Nach dem Aufbau des Baumes mit der Trainingsmenge:
  - Versuche, zwei Blattknoten durch Löschen eines Tests zusammenzulegen,
  - Solange dadurch die Fehlerrate auf der Pruningmenge verringert wird.

Umwandlung von Bäumen in Regeln

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

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

Umwandlung von Bäumen in Regeln



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

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

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

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

$R_5 = Wenn\ (Ausblick = regnerisch) \wedge (Wind = schwach), dann\ Tennis\ spielen$