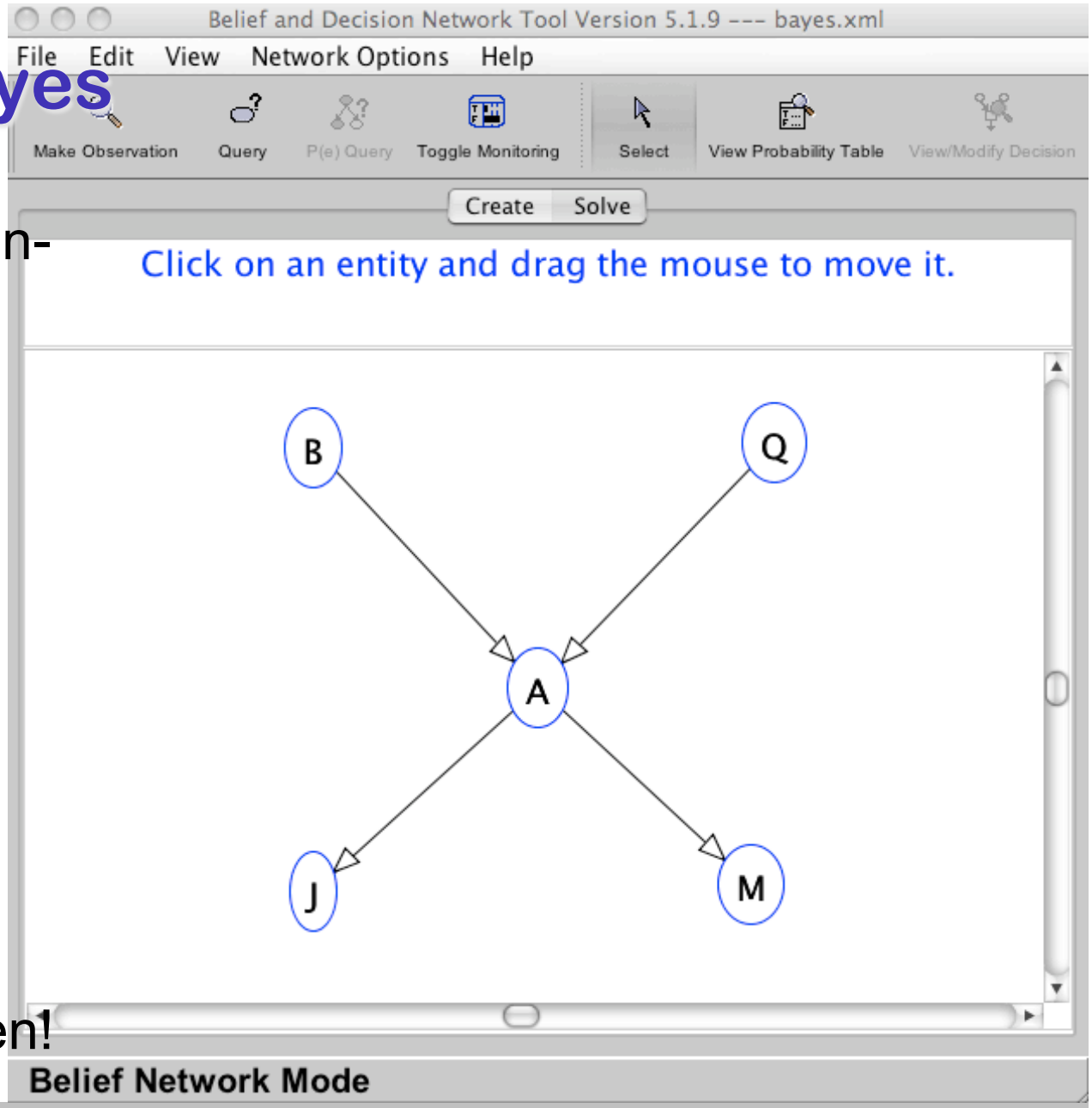
aispace.org/bayes

- Tool zum Experimentieren mit kleinen Bayes-Netzen
- Modus 1: Erstellen
- Modus 2: Rechnen
- Möglichkeiten zum Abfragen und Monitoring von Variablen
- Möglichkeit zum Einfügen von „Beobachtungen“
- siehe auch Übungen!

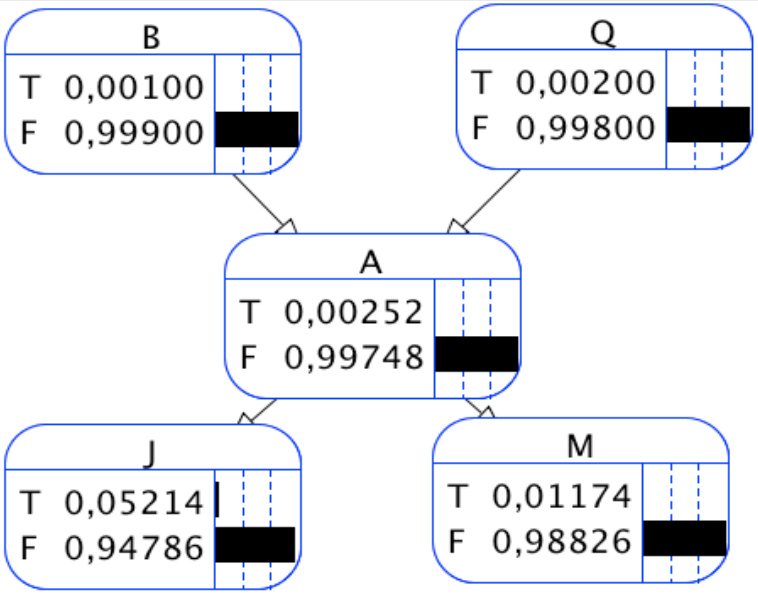


Beispiel, Fortsetzung

Verteilungen a priori

… gemäß Definition des Netzes

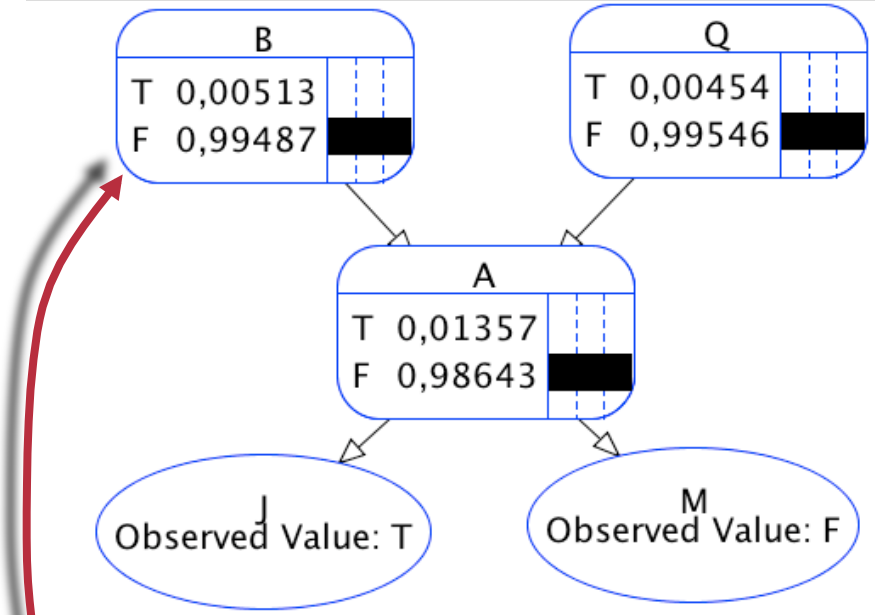
Click on a node to stop or start monitoring its probability.



Verteilungen a posteriori

… nach Beobachtungen $j, \neg m$

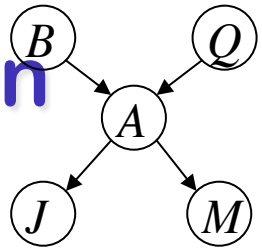
Click on a node to make an observation about its value.



Gesuchte Verteilung $\mathbf{P}(B \mid j, \neg m)$!

Und wie man die ausrechnet, …

Bspl. Forts.: Inferenz durch Ausrechnen



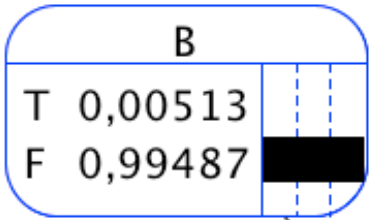
$$\mathbf{P}(B \mid j,\text{-}m)$$
$$= \mathbf{P}(B, j,\text{-}m) \,/\, \mathbf{P}(j,\text{-}m) = \mathbf{P}(B, j,\text{-}m) \,/\, P(j,\text{-}m) \quad \text{Produktregel}$$
$$= \alpha\, \mathbf{P}(B, j,\text{-}m) \quad \text{Normalisierung mit Konstante}$$
$$= \alpha \sum_q \sum_a [\mathbf{P}(B, Q{=}q, A{=}a, j, \text{-}m)] \quad \text{Konditionalisierung}$$

Weiter mit ***B=b*** (anderer Fall ***B=-b*** entsprechend)

$$= \alpha \sum_q \sum_a [P(b)P(Q{=}q)P(A{=}a|b,q)P(j|a)P(\text{-}m|a)] \quad \text{bed. Unabhängigk.}$$
$$= \alpha\, P(b) \sum_q P(Q{=}q) \left[\sum_a P(A{=}a|b,q)P(j|a)P(\text{-}m|a)\right]$$

Durch Einsetzen der W'keiten gemäß Bayes-Netz:

$$\mathbf{P}(B \mid j,\text{-}m) \approx \alpha \langle 0.0002568, 0.0497979 \rangle$$
$$\approx \langle 0.00513, 0.99487 \rangle$$



$\alpha \approx 20$

➥ Die W'keit für einen Einbruch ist von a priori 0,1% auf 0,513% gestiegen

Inferenz durch vollständiges Ausrechnen

function ENUMERATION-ASK(X, e, bn) **returns** a distribution over X
  inputs: X, the query variable
    e, observed values for variables $\mathbf{E}$
    bn, a Bayesian network with variables $\{X\} \cup \mathbf{E} \cup \mathbf{Y}$

  $\mathbf{Q}(X) \leftarrow$ a distribution over X, initially empty
  for each value x_i of X **do**
    extend e with value x_i for X
    $\mathbf{Q}(x_i) \leftarrow$ ENUMERATE-ALL(VARS[bn], e)
  return NORMALIZE($\mathbf{Q}(X)$)

versteckte Variablen

$\{X\} \cup \mathbf{E} \cup \mathbf{Y}$

function ENUMERATE-ALL($vars$, e) **returns** a real number
  if EMPTY?($vars$) **then return** 1.0
  $Y \leftarrow$ FIRST($vars$)
  if Y has value y in e
    then return $P(y \mid Pa(Y)) \times$ ENUMERATE-ALL(REST($vars$), e)
    else return $\Sigma_y \; P(y \mid Pa(Y)) \times$ ENUMERATE-ALL(REST($vars$), $\mathbf{e}_y$)
      where $\mathbf{e}_y$ is e extended with $Y = y$

Parents, direkte Vorgänger von Y

Eigenschaften des vollständigen Ausrechnens

- ☺ Terminiert sicher mit korrektem Ergebnis
- ☺ Speicherbedarf: $O(n)$ (n ZV, „Tiefentraversierung“)
- ☹ Zeit: $O(m^n)$ für diskrete ZV max. m Werte (boolesche ZV: m=2)
- ☹ **Satz**: Exakte Inferenz in Bayes-Netzen ist NP-vollständig
  Beweisidee: Abbildung propositionaler Inferenz in Bayes-Netz-Inferenz

Optimierungsmöglichkeiten

- *caching* mehrfach auftauchender Teilergebnisse (s.Russell/N.)
- Überführung von Bayes-Netzen in Polybäume (s. Russell/N.)
- Variablenelimination (s. Russell/N.)

Bayes-Netz-Inferenz durch Approximation

Rechne (bedingte) Verteilungen nicht aus,
sondern ermittle sie durch Stichproben aus dem Netz!

function PRIOR-SAMPLE(bn) **returns** an event sampled from bn
  inputs: bn, a belief network specifying joint distribution $\mathbf{P}(X_1, \ldots, X_n)$

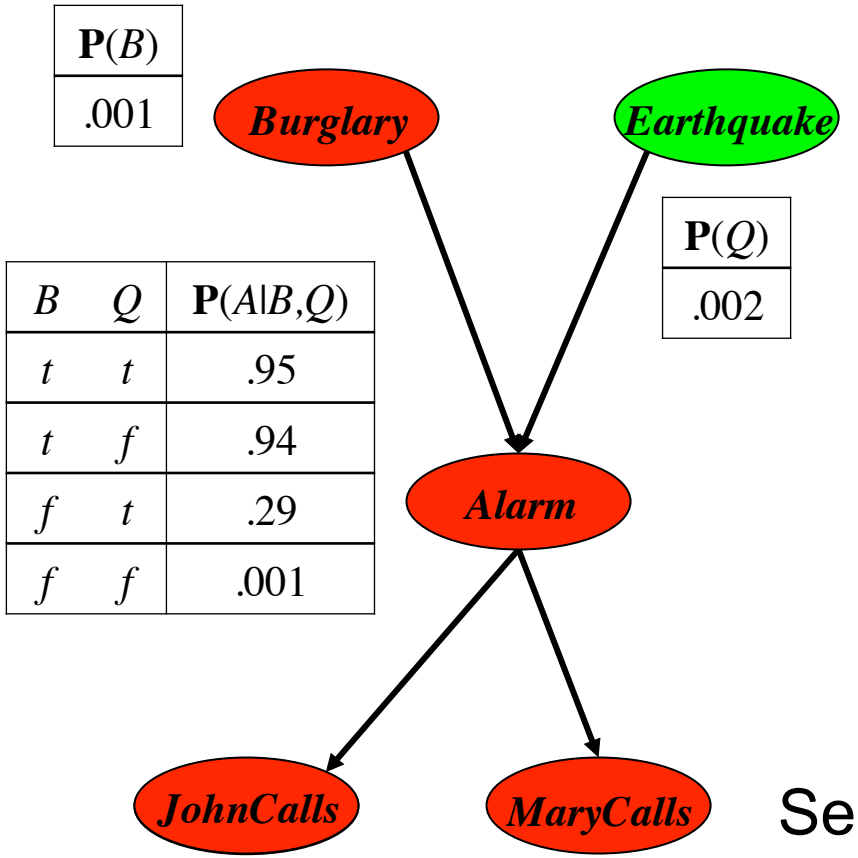
  $\mathbf{x} \leftarrow$ an event with n elements
  for $i = 1$ **to** n **do**
    $x_i \leftarrow$ a random sample from $\mathbf{P}(X_i \mid Parents(X_i))$
  return $\mathbf{x}$

W‘keit, dass Funktion Stichprobe (Ereignis) $(x_1, \ldots, x_n)$ erzeugt:

$$S_{PS}(x_1, \ldots, x_n) = \prod_i P(x_i \mid Parents(X_i)) = P(x_1, \ldots, x_n)$$

d.h.: hinreichend viele Stichproben approximieren Verteilung!

Beispiel: Alarmanlage – jetzt approximativ



1. Ziehe B nach $\mathbf{P}(B)=\langle 0.001, 0.999\rangle$
   ➥ Ergebnis sei $B=false$
2. Ziehe Q nach $\mathbf{P}(Q)=\langle 0.002, 0.998\rangle$
   ➥ Ergebnis sei $Q=true$
3. Ziehe A nach $\mathbf{P}(A|\text{-}b,q)=\langle 0.29, 0.71\rangle$
   ➥ Ergebnis sei $A=false$
4. Ziehe J nach $\mathbf{P}(J|\text{-}a)=\langle 0.05, 0.95\rangle$
   ➥ Ergebnis sei $J=false$
5. Ziehe M nach $\mathbf{P}(M|\text{-}a)=\langle 0.01, 0.99\rangle$
   ➥ Ergebnis sei $M=false$

Sei $N_{PS}(x_1, \ldots, x_n)$ die Häufigkeit, mit der $(x_1, \ldots, x_n)$ gezogen wurde.

Nach N (hinreichend groß) Stichproben:

$$P(x_1, \ldots, x_n) \approx N_{PS}(x_1, \ldots, x_n) / N$$

Grenzen&Verbesserung von PRIOR-SAMPLE

- Vorhandene Evidenz wird nicht verwendet:
  $\mathbf{P}(X)$ wird approximiert, aber nicht ohne Weiteres $\mathbf{P}(X|\mathbf{e})$
- Bei vielen Variablen oder unwahrscheinlicher Evidenz $\mathbf{e}$
  (kleines $P(\mathbf{e})$) werden viele „Nieten“ gezogen
  ↪ N muss sehr groß sein
- Je mehr Evidenzvariable, desto kleiner $P(\mathbf{e})$
  ↪ Algorithmus wird relativ schlechter bei mehr Information!

Verbesserung: **Gewichtete Stichproben** (WEIGHTED-SAMPLE)

- Belege Evidenz-Variablen wie durch $\mathbf{e}$ vorgegeben
- Wichte&normalisiere gezogene Verteilung nach $\mathbf{P}(\mathbf{e}|X,\mathbf{e})$
- Verbessert PRIOR-SAMPLE; großes $\mathbf{e}$ kann weiter Probleme machen! (besonders wenn Evid.-Variablen „spät“ in der ZV-Reihenfolge)

Was hat uns Kapitel 4 gelehrt?

- Wissensrepräsentationsformalismen suchen Kompromiss zwischen Ausdrucksstärke („reichhaltig“), Repräsentations- und Inferenz-Effizienz („kompakt und flink“), und Lücken- und Fehlertoleranz („robust“)
- Zudem sollen sie formal fundiert sein (z.B. Logik, W‘theorie)
- ft kann Unsicherheit „summarisch“ in Form von W‘keitsinformation repräsentiert werden
- Bayes-Netze behandeln effizient Unsicherheit in Termini von W‘keit (durch Ausnutzen bedingter Unabhängigkeit!)

5. Maschinelles Lernen

1. Was ist KI?
2. Logik und Inferenz
3. Suche als Problemlösen
4. Schließen unter Unsicherheit
5. **Maschinelles Lernen**
6. Ausblick: „Rationale“ Roboter

- 5.1 Überwachte Lernverfahren
- 5.2 Unüberwachte Lernverfahren
- 5.3 Vertiefung: Neuronale Netze
- 5.4 Verstärkungslernen

Was ist Lernen?

… für Computer ist nicht Behalten/Speichern das Problem

… sondern selbständiger Transfer von früherer „Erfahrung“

„… changes in the system that are adaptive in the sense that they enable the system to do the same task or tasks drawn from the same population more efficiently next time.“
Herbert Simon

… zwar sind darin viele Punkte unklar:

- wer macht die „changes“, darf das ein Programmierer sein?
- was ist „the same task“? was ist „the population“?
- was heißt „more efficiently“?
- warum eigentlich erst „next time“ — nicht schon für „first time“?

… doch wirklich gute Definition gibt es nicht! Ertel:

Definition 8.1 *Ein Agent heißt lernfähig, wenn sich seine Leistungsfähigkeit auf neuen, unbekannten Daten, im Laufe der Zeit (nachdem er viele Trainingsbeispiele gesehen hat) verbessert (gemessen auf einem geeigneten Maßstab).*

Varianten von Lernen

- **Überwachtes Lernen**: Gegeben Paare $\langle \boldsymbol{In}, Erwartet\_Out \rangle$, leite Fkt. f ab, sodass für neue Eingaben gilt $f(\boldsymbol{In}')=Erwartet\_Out'$
  Beispiel: Lerne Vorhersage für Wechselkurs $≈€, gegeben Wirtschaftsdaten und Wechselkurse aus der Vergangenheit

- **Unüberwachtes Lernen**: Gegeben Merkmalvektoren $\langle \boldsymbol{In} \rangle$, leite Fkt. f ab, die Regularitäten/Einteilungen beschreibt
  Beispiel: Finde potenziell „interessante“ Muster in riesengroßen Datenbeständen (➥ **Data Mining**), z.B. in Kundendaten von PayBack

- **Reinforcement-Lernen**: Gegeben Tripel $\langle \boldsymbol{In}, f:\boldsymbol{In}\rightarrow\boldsymbol{Out}, Reward \rangle$, modifiziere Fkt. f, sodass in Zukunft $Reward$ durch $f'(\boldsymbol{In})$ optimiert wird (Jargon: $Reward$➥**Reinforcement**-Signal)
  Beispiel: Lerne die Koordination der Motorsignale einer sechsbeinigen Laufmaschine für „glatte“, effiziente Schreitbewegung in Richtung eines Zielpunkts ($Reward$ z.B.: Näherung z. Ziel – Zeit-Integral der Nick/Rollwinkel)