Bayes'sche Netze: Konstruktion, Inferenz, Lernen und Kausalität

Volker Tresp

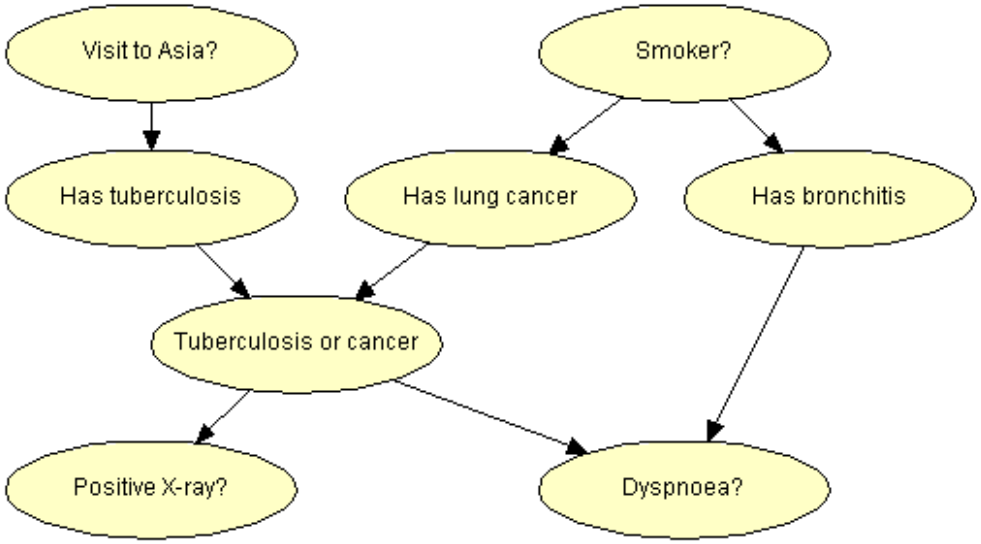
Einführung

- Bisher haben wir uns fast ausschließich mit überwachtem Lernen beschäftigt: Ziel war es, eine (oder mehr als eine) Zielgröße basierend auf Informationen über die Eingangsgrößen abzuschätzen
- Wenn anwendbar, ist überwachtes Lernen auch oft die beste Lösung und es stehen mächige Werkzeuge wie Neuronale Netze oder die SVM zur Verfügung
- Es gibt jedoch auch Problemstellungen, bei denen überwachtes Lernen nicht anwendbar ist, insbesondere wenn man mit fehlenden Daten oder unsicherer Information umgehen muss und wenn man nicht nur an einer Zielgrösse sondern an einer Vielzahl von Zielgrößen interessiert ist
- Ein Beispiel ist eine medizinisch diagnostische Domäne, die aus einer Vielzahl von Krankheiten, Symptomen und Kontextinformationen bestehen kann: Über einen gegebenen Patienten sind nur wenige Informationen bekannt und man interessiert sich für die Vorhersage der Wahrscheinlichkeit einer Vielzahl von Krankheiten
- Dies sind einige der Gründe, weshalb Bayes'sche Netze in den letzten Jahren viel Aufmerksamkeit erfahren haben

Bayes'sche Netze

- Bayes'sche Netze konstruieren eine hochdimensionale W.-Verteilung basierend auf lokalen probabilistischen Regeln
- Sie sind eng verknüpft mit einer kausalen Modellierung der Welt
- Regelbasierte Systeme wahren der dominierende Ansatz zu Wissensmodellierung zu den Hochzeiten der Künstlichen Intelligenz (KI); die Fokussierung auf Logik in regelbasierten Systemen war den Problemen der Realwelt jedoch nicht angemessen: In der Mehrzahl der Anwendungen ist der Umgang mit Unsicherheit und mit fehlenden Informationen essentiell; eben dafür sind logikbasierte regelbasierte Systeme nur unzureichend ausgelegt
- Bayes'sche Netze begannen als ein kleines Randgebiet und haben sich in den 90er Jahren zu einem der wichtigsten Ansätze in der KI entwickelt
- Sie stellen ebenso flexible und mächtige Modelle zur Realisierung lernender Systeme dar

Chest Clinic



Definition Bayes'scher Netze

- Die Variablen einer Domäne werden zu Knoten in einer entsprechenden graphischen Repräsentation
- Gerichtete Kanten (Pfeile) repräsentieren direkte kausale Abhängigkeiten von Elternknoten zu Kindernoten
- Quantitative Spezifikation der Abhängigkeiten:
  - Für Knoten ohne Eltern müssen a prior Wahrscheinlichkeiten definiert werden

$$P(A = i) \quad \forall i$$

  - Für Knoten mit Eltern müssen bedingte Wahrscheinlichkeiten spezifiziert werden

$$P(A = i | B = j, C = k) \quad \forall i, j, k$$

Gemeinsame W.-Verteilung

- Ein Bayes'sches Netz spezifiziert eine W-Verteilung in der Form

$$P(X_1, \ldots X_M) = \prod_{i=1}^{M} P(X_i | \mathsf{par}(X_i))$$

wobei $\mathsf{par}(X_i)$ die Menge der Elternknoten angibt (diese Menge kann leer sein)

Definition eines Bayes'schen Netzes aus der Produktzerlegung von W.-Verteilungen

- Wir beginnen mit der Produktzerlegung einer W.-Verteilung (siehe Review über W.-Theorie)

$$P(X_1, \ldots X_M) = \prod_{i=1}^{M} P(X_i | X_1, \ldots, X_{i-1})$$

- Diese Zerlegung ist immer in jeder beliebigen Reihenfolge der Variablen möglich; jeder Term enthält die bedingte W.-Verteilung eines Knoten bedingt auf alle Vorgänger des Knotens

- Die Abhängigkeiten reduzieren sich, wenn eine Variable nicht von allen Vorgängerknoten abhängt

$$P(X_i | X_1, \ldots, X_{i-1}) = P(X_i | \mathrm{par}(X_i))$$

mit

$$\mathrm{par}(X_i) \subseteq X_1, \ldots, X_{i-1}$$

- Wenn die Ordnung der Variablen der kausalen Ordnung entspricht, erhalten wir die kausale Zerlegung wie zuvor
- Eine Zerlegung ist jedoch in jeder beliebigen Reihenfolge der Variablen möglich; allerdings liefert eine Zerlegung, die die kausale Ordnung berücksichtigt, die am leichtesten interpretierbare und die sparsamste Repräsentation mit den wenigsten Kanten.

Inferenz

- Die wichtigste Operation in Bayes'schen Netzen ist die Inferenz: gegeben, dass eine Menge von Variablen bekannt ist, so ist man an der Verteilung einer (oder mehrerer) der unbekannten Variablen interessiert
- Seien $\mathcal{X}^m$ die Menge der bekannten (gemessenen) Variablen und sei X^q die Variable, an deren a posteriori Verteilung wir interessiert sind. Die restlichen Variablen seien $\mathcal{X}^r$.
- Im einfachen Inferenzansatz geht man folgendermaßen vor:
  - Marginalisierung: die restlichen Variablen sind ohne Interesse und müssen herausintegriert werden

$$P(X^q, \mathcal{X}^m) = \sum_{\mathcal{X}^r} P(X_1, \ldots X_M)$$

  - Berechnung der Normalisierung:

$$P(\mathcal{X}^m) = \sum_{\mathcal{X}^q} P(X^q, \mathcal{X}^m)$$

– Berechnung der bedingten Wahrscheinlichkeit

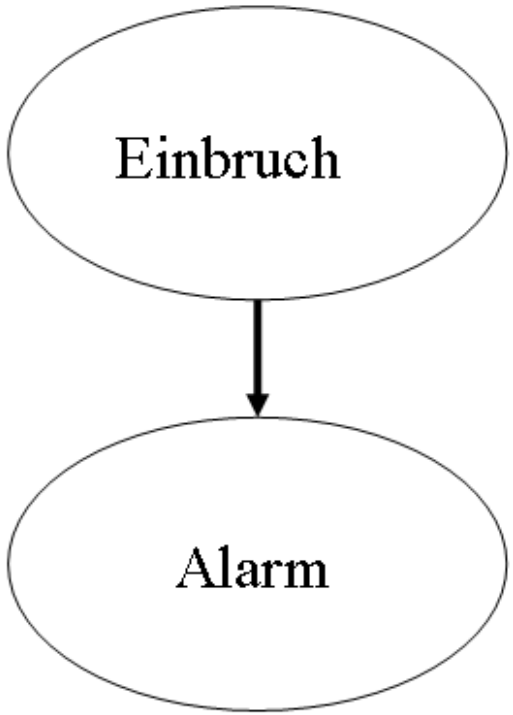
$$P(X^q|\mathcal{X}^m) = \frac{P(X^q, \mathcal{X}^m)}{P(\mathcal{X}^m)}$$

– Für die Inferenz ist die Berechnung der Normalisierung in vielen Fällen nicht notwendig; Im Lernen von Bayes Netzen spielt sie allerdings eine wichtige Rolle

Inferenz in einem einfachen Bayes Netz

$$P(Einbruch = 1) = 0.01$$
$$P(Einbruch = 0) = 0.99$$

$$P(Alarm \mid Einbruch) = 0.98$$
$$P(Alarm \mid \neg Einbruch) = 0.01$$



Marginalisierung (hier unnötig):

$$P(A,E) = P(A \mid E)P(E) = 0.98 \times 0.01 = 0.0098$$

Normalisierung:

$$P(A) = P(A \mid E)P(E) + P(A \mid \neg E)P(\neg E)$$
$$= 0.98 \times 0.01 + 0.01 \times 0.99 = 0.0197$$

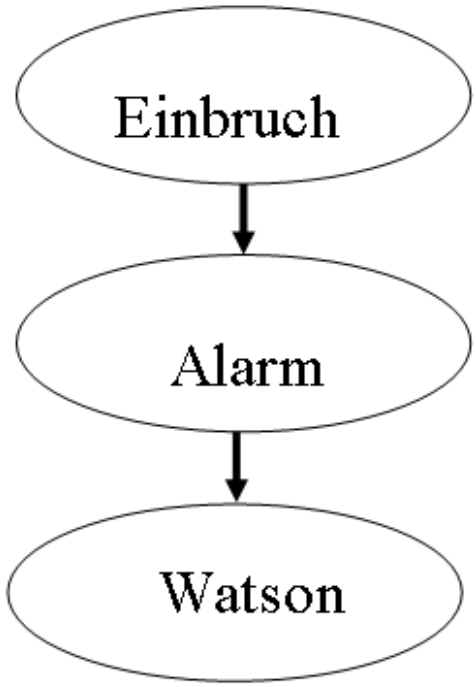
Konditionierung:

$$P(E \mid A) = \frac{P(A,E)}{P(E)} = \frac{0.0098}{0.0197} = 0.49$$

Watson meldet Alarm, neigt jedoch zu Scherzen

$$P(WatsonRuftAn \mid Alarm) = 0.7$$

$$P(WatsonRufAn \mid \neg Alarm) = 0.1$$



Gemeinsame Verteilung:

$$P(W \mid A)P(A \mid E)P(E)$$

Marginalisierung:

$$P(W, E) = P(W \mid A)P(A \mid E)P(E) + P(W \mid \neg A)P(\neg A \mid E)P(E)$$

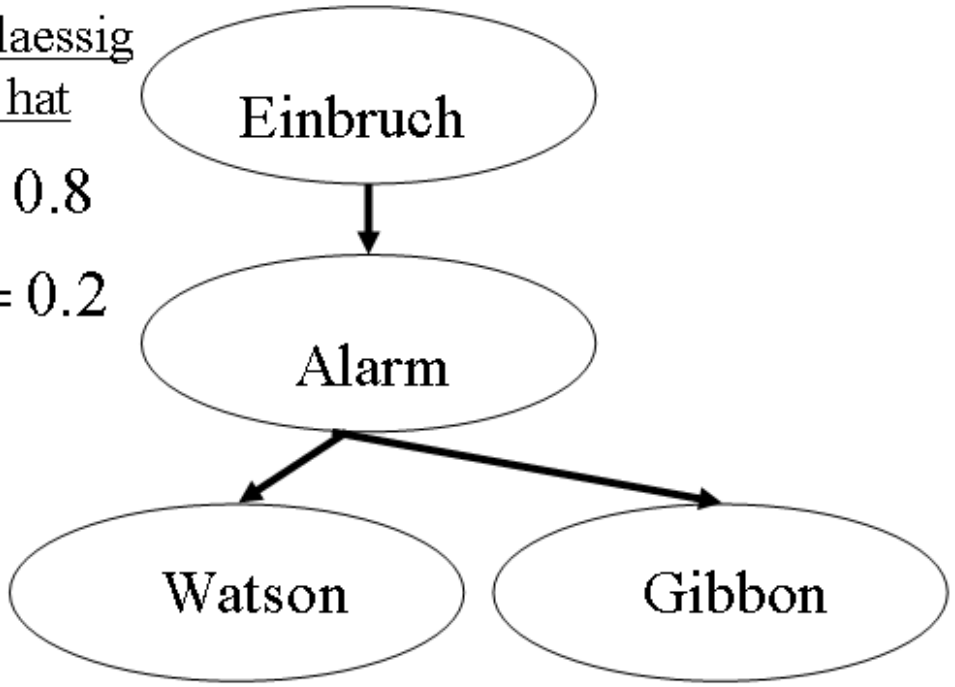
Normalisierung: $P(W) = P(E,W) + P(\neg E,W)$

Konditionierung: $P(E \mid W) = \frac{P(E,W)}{P(W)} = 0.0615$

Holmes ruft Frau Gibbon an, die zuverlaessig ist, jedoch ein kleines Alkoholproblem hat

$$P(GibbonBestaetigt \mid Alarm) = 0.8$$

$$P(GibbonBestaetigt \mid \neg Alarm) = 0.2$$



Gemeinsame Verteilung:

$$P(G \mid A)P(W \mid A)P(A \mid E)P(E)$$

Nach dem gleichen Rezept::

Konditionierung:

$$P(E \mid G) = \frac{P(E,G)}{P(G)} = 0.0372$$

$$P(E \mid G,W) = \frac{P(E,G,W)}{P(G,W)} = 0.179$$

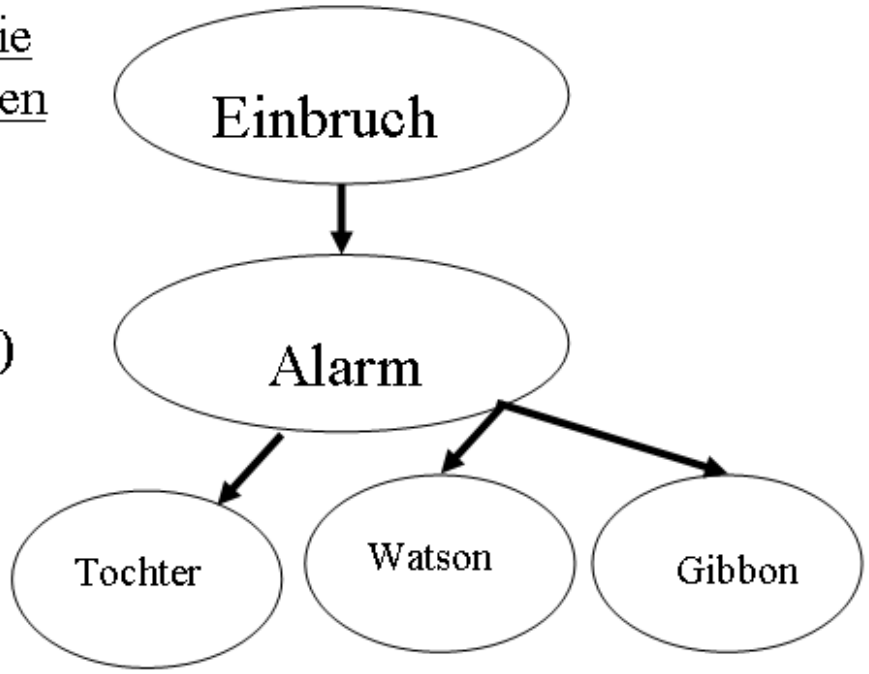
Holmes hat eine zuverlässige Tochter, die anrufen würde, wenn Sie den Alarm hören würde

$$P(TochterRuftAn \mid Alarm) = 0.7$$

$$P(TochterRufAn \mid \neg Alarm) = 0.0$$

Gemeinsame Verteilung:

$$P(G \mid A)P(W \mid A)P(A \mid E)P(E)$$



Nach dem gleichen Rezept::

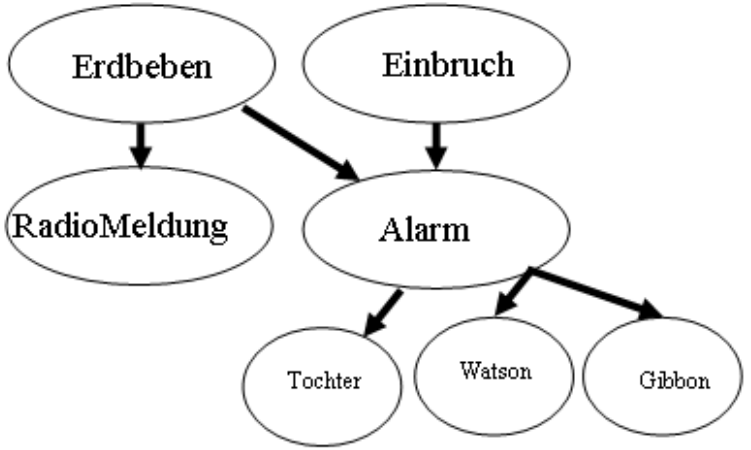
Konditionierung: $$P(E \mid T) = \frac{P(E,T)}{P(T)} = 0.49$$

Weil: $$P(A \mid T) = \frac{P(A,T)}{P(T)} = 1$$

Holmes hoert im Radio, dass es gerade ein Erdbeben gegeben hat

$$P(Erdbeben) = 0.0001 \qquad P(Radiomeldung \mid Erdbeben) = 0.4$$

$$P(Alarm \mid Erdbeben, \neg Einbruch) = 0.20$$
$$P(Alarm \mid \neg Erdbeben, Einbruch) = 0.98$$
$$P(Alarm \mid Erdbeben, Einbruch) = 0.9840$$
$$P(Alarm \mid \neg Erdbeben, \neg Einbruch) = 0.01$$



Gemeinsame Verteilung:

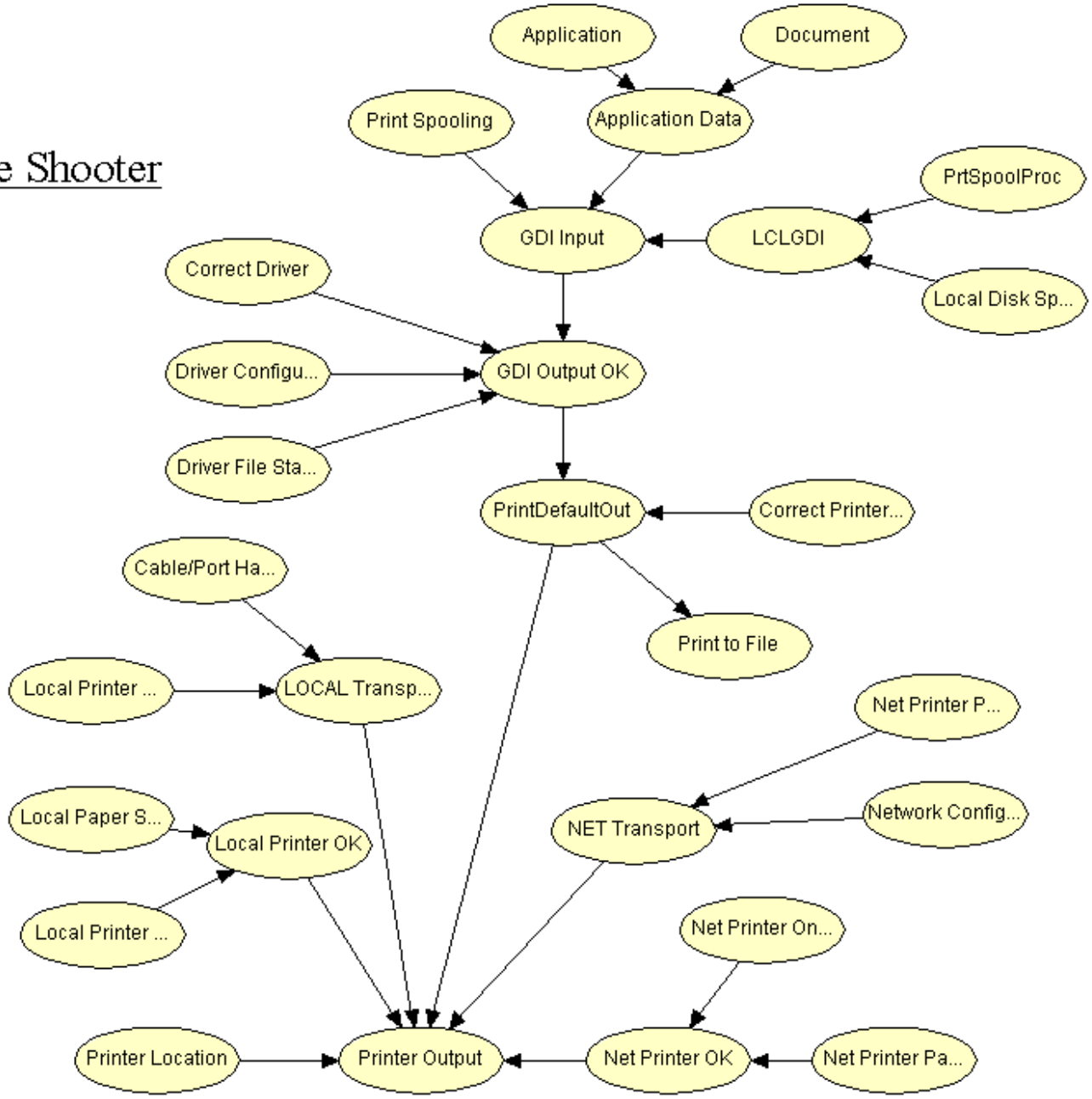
$$P(G \mid A)P(W \mid A)P(A \mid E, Erd)P(E)P(Erd)P(R \mid Erd)$$

$$P(E \mid T) = 0.49 \qquad P(E \mid T, R) = 0.0473$$

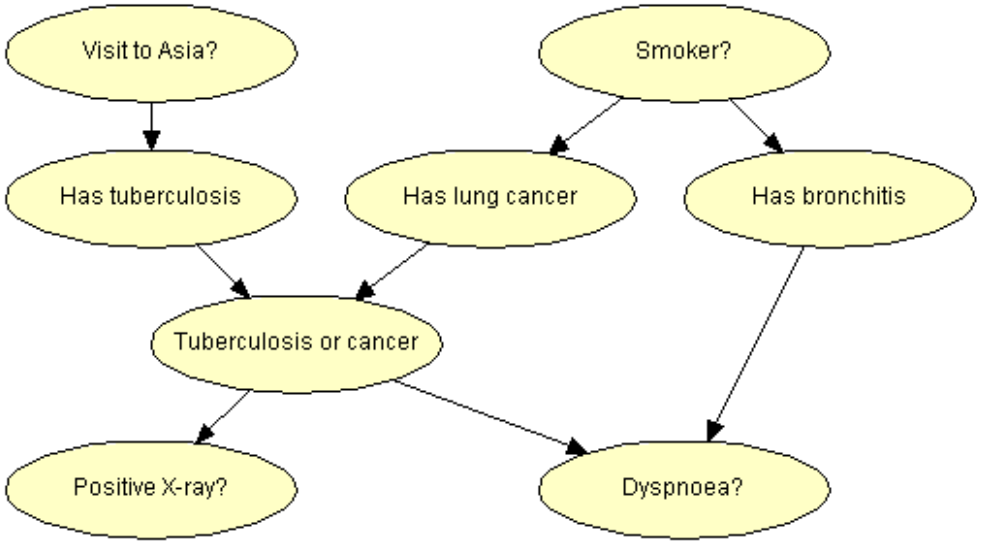
Explaining away: da das unwahrscheinliche Erdbeben den Alarm erklärt, wird der Einbruch wieder unwahrscheinlicher!

Printer Trouble Shooter

(Microsoft)



Chest Clinic



A real Bayes net: Alarm

Domain: Monitoring Intensive-Care Patients

- 37 variables
- 509 parameters

  …instead of 2^{37}

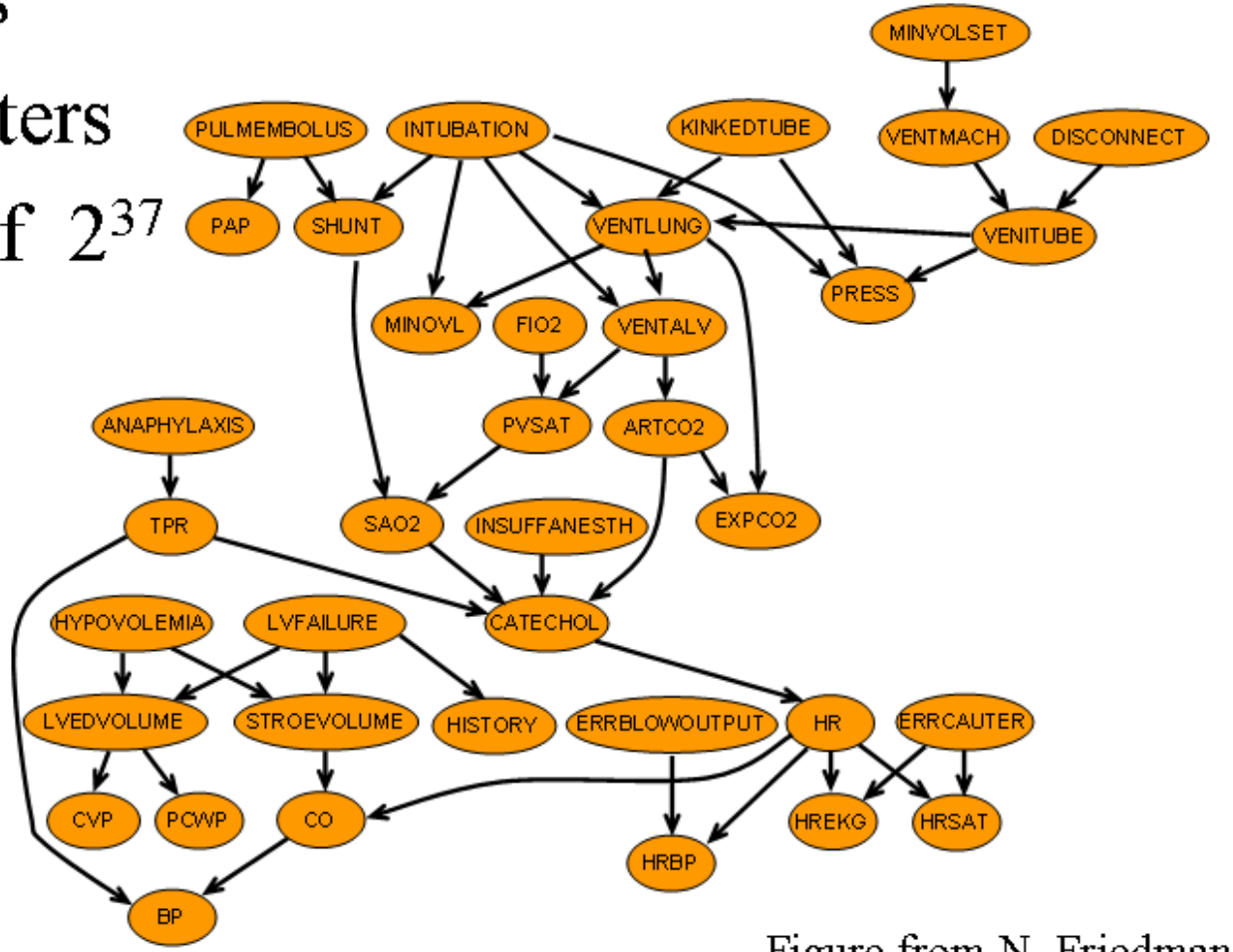
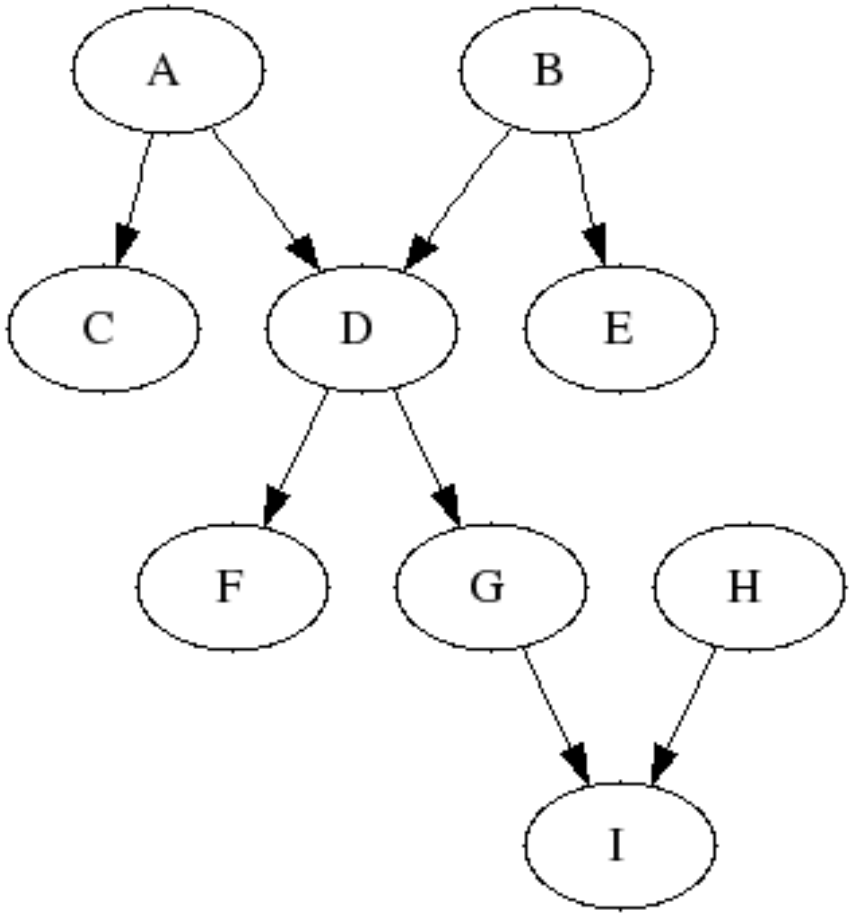


Figure from N. Friedman

Inferenz in BNs ohne Zyklen

- Durch Konstruktion gibt es keine Zyklen im gerichteten Bayes Netz; die Struktur der Bayes Netze wird deshalb auch als *directed acyclic graph* (DAG) bezeichnet
- Es kann jedoch Zyklen geben, wenn man die Pfeilrichtungen ignoriert
- Wir betrachten zunächst nur Bayes Netze, die auch keine Zyklen enthalten, wenn man die Pfeilrichtungen wegläßt; die Struktur des Bayes Netzes ist dann ein Poly-tree: es gibt höchstens einen gerichteten Weg zwischen zwei Knoten



- Die teure Operation in der Inferenz ist die Marginalisierung, denn sie Summe enthält exponentiell viele Terme
- Ziel ist es nun, die Struktur des Netzes auszunützen, um eine effiziente Marginalisierung zu erreichen

- Beispiel: betrachte die Markov Kette

$$P(X_1)P(X_2|X_1)\dots P(X_M|X_{M-1})$$

Dann ist $P(X_1)$ bekannt und muss schon einmal nicht berechnet werden. Weiterhin ist

$$P(X_M) = \sum_{X_1,\dots,X_{M-1}} P(X_1)P(X_2|X_1)\dots P(X_M|X_{M-1})$$

$$= \sum_{X_{M-1}} P(X_M|X_{M-1}) \sum_{X_{M-2}} P(X_{M-1}|X_{M-2}) \dots \sum_{X_1} P(X_2|X_1)P(X_1)$$

Oben muss man 2^{M-1} Terme summieren, unten nur noch $2(M-1)$.

- Im allgemeinen kann Inferenz in Form eines Message Passing Algorithmus effizient implementiert werden

Belief Propagation (Sum-Product-Rule)

- Update für Knoten X mit Eltern $U_1, \ldots, U_n$ und Kindern $Y_1, \ldots, Y_m$ und lokaler Verteilung $P(x|u_1, \ldots, u_n)$ (Notation nach Pearl: Probabilistic Reasoning)
- Im Netzwerk werden Botschaften (messages) in Pfeilrichtung (Top-down) und entgegen der Pfeilrichtung (Bottom-up) propagiert
- $\pi_{Y_j}(x)$ ist die Top-down message von Knoten x zu seinem Kindknoten Y_j
- $\lambda_X(u_i)$ ist die Bottom-up message von Knoten x zu seinem Elternknoten u_i

Propagationsregeln

- In Gleichungen berechnen sich diese Botschaften für Knoten X lokal zu
- Top-down propagation (für alle Zustände x):

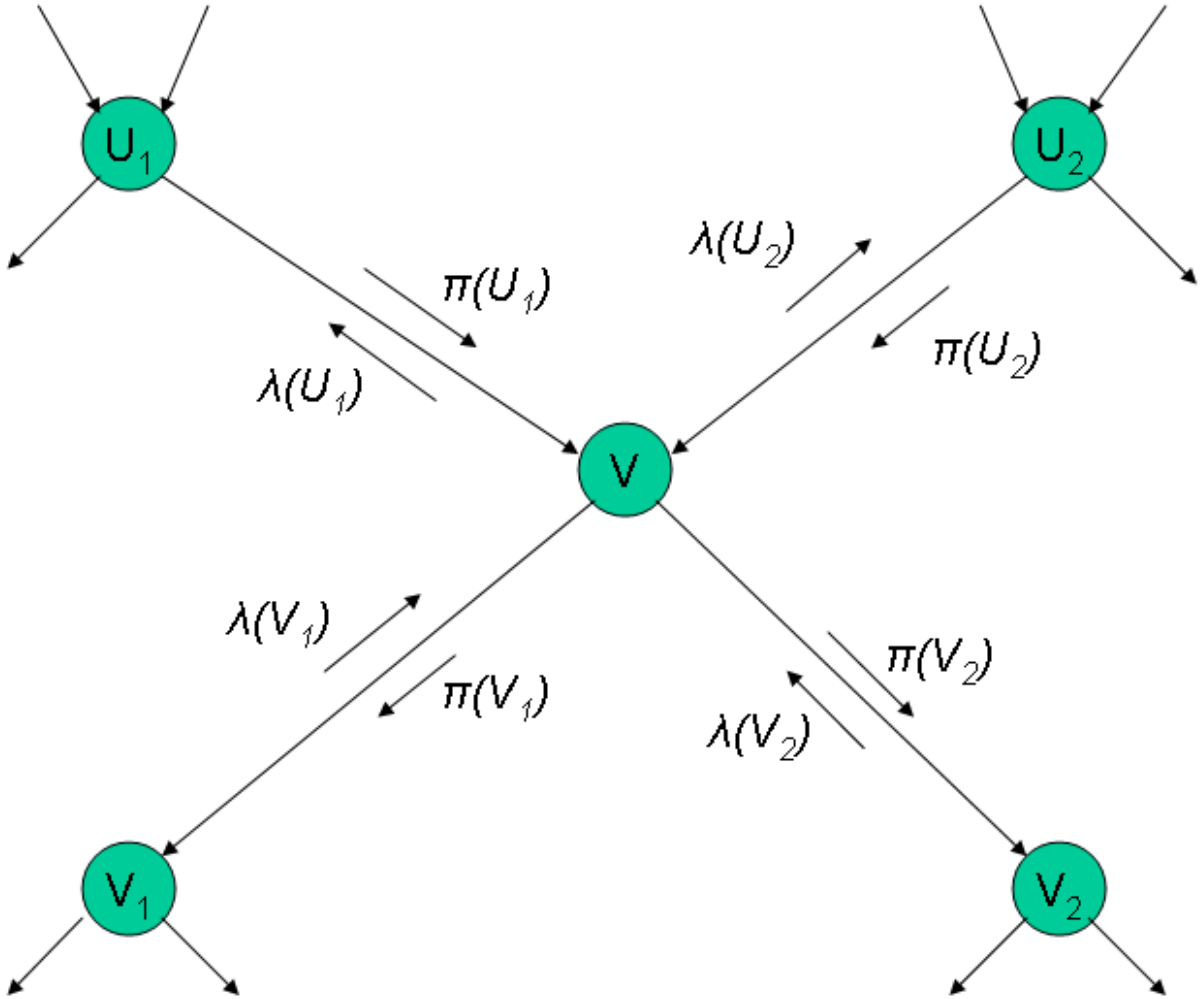
$$\pi_{Y_j}(x) = \left[\prod_{k \neq j} \lambda_{Y_k}(x)\right] \sum_{u_1,\ldots,u_n} P(x|u_1,\ldots,u_n) \prod_i \pi_X(u_i)$$

- Bottom-up propagation (für alle Zustände x):

$$\lambda_X(u_i) = \sum_x \left[\lambda(x) \sum_{u_k : k \neq i} P(x|u_1,\ldots,u_n) \prod_{k \neq i} \pi_X(u_k)\right]$$

Mit $\lambda(x) = \prod_j \lambda_{Y_j}(x)$

Pearl's Belief Propagation

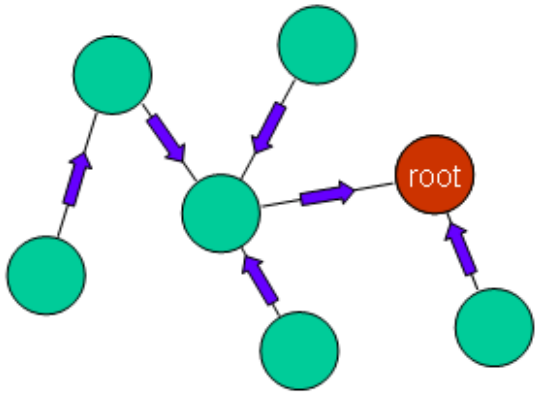


Belief Propagation

aka Pearl's algorithm, sum-product algorithm

Generalization of forwards-backwards algo. /RTS smoother from chains to trees - linear time, two-pass algorithm

Collect



Distribute

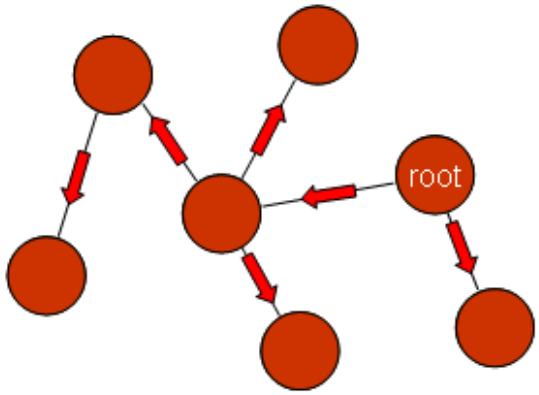


Figure from P. Green

Berechnung des Beliefs

- Es reichen zwei Passagen durch das Netz aus, um alle Knoten zu berechnen
- Mit

$$\pi(x) = \sum_{u_1,\ldots,u_n} P(x|u_1,\ldots,u_n) \prod_i \pi_X(u_i)$$

- Ist (Z normalisiert die Verteilung)

$$Bel(x) = P(x|Evidence) = \frac{1}{Z}\lambda(x)\pi(x)$$

Initialisierung

- Für Knoten mit bekanntem Zustand: $\pi(x) = 1$, $\lambda(x) = 1$
- Für Knoten mit unbekanntem Zustand:: $\pi(x) = 0$, $\lambda(x) = 0$

Ausnahmen:

- Für Knoten ohne Eltern: $\pi(x) = p(x)$
- Für Knoten ohne Kinder: $\lambda(x) = 1$

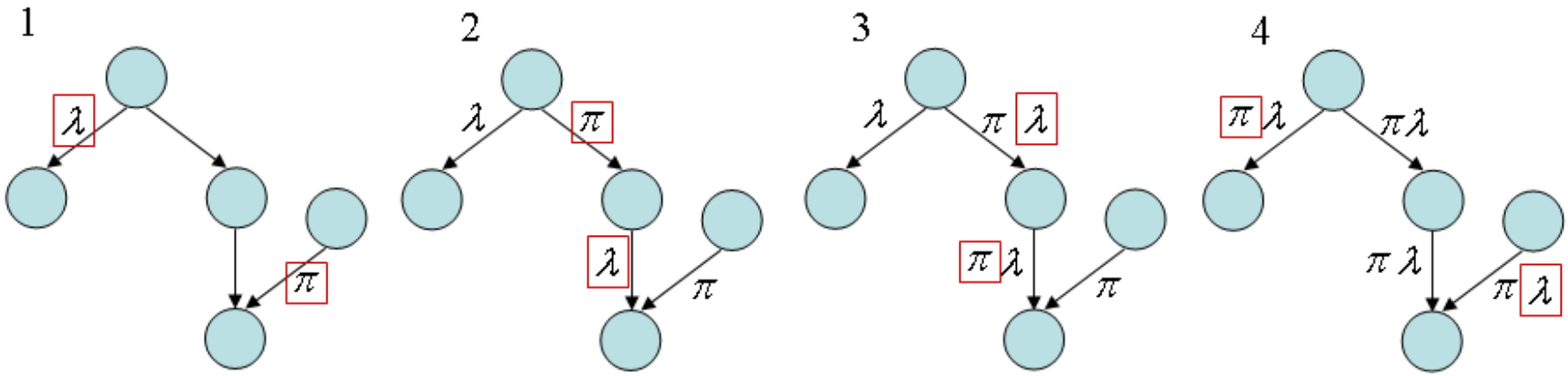
Reihenfolge

- Prinzipiell können Knoten in beliebiger Reihenfolge upgedated werden

  Effizient sind die Anwendung folgender Regeln:

- Wenn Messages von allen Eltern vorliegen, berechne $\pi(x)$
- Wenn Messages von allen Kindern vorliegen, berechne $\lambda(x)$
- Wenn Messages von allen Eltern vorliegen, und allen Kindern, außer möglicherweise Y_i, berechne $\pi_{Y_i}(x)$
- Wenn Messages von allen Kindern vorliegen, und allen Eltern, außer möglicherweise U_i, berechne $\lambda_{U_i}(x)$

Beispiel: Propagation von Evidenz



Max-Propagation

- Ähnlich effizient kann die maximal-wahrscheinliche Konfiguration berechnet werden (Max-product Rule)

Belief Propagation in anderer Form

- Hidden Markov Modelle (HMMs) sind die Basis moderner Spracherkennungssysteme; sie können als Bayes'sches Modell verstanden werden
- Die Berechnung der wahrscheinlichsten Zustände, gegeben die akustische Messungen, wird mit dem Viterbi Algorithmus berechnet; dieser ist equivalent zum Max-Prop Algorithmus
- Der Kalman Filter ist equivalent zum HMM Modell, nur dass die Variablen stetig sind und die bedingten Wahrscheinlichkeiten Gaussverteilt sind; die Schätzung der versteckten Zustände wird mit einem Belief Propagation Algorithmus berechnet

Belief Propagation mit Schleifen

- Mit Schleifen im Graphen ohne Pfeilen ist der Algorithmus eigentlich so nicht anwendbar, denn die Informationen aus verschiedenen Richtungen kann korreliert sein
- Dennoch wird er oft angewandt und konvergiert zu guten Lösungen; hier müssen wiederholt message passing angewandt werden, bis eine Konvergenz erreicht wird (was nicht immer der Fall ist)
- Die Anwendung des Belief Propagation auf Netze mit Schleifen wird als *Loopy Belief Propagation* bezeichnet
- Eine korrekte Form der Implementierung ist der *junction tree algorithm* ; allerdings ist er nur effizient, wenn die Schleifen im BN nicht zu gross sind (in meisten Entwicklungsumgebungen ist der *junction tree algorithm* implementiert)
- Interessant ist, dass die Dekodierung im Turbocode und in Low-Density Parity-Check (LDPC) Codes über loopy Beliefpropagation geschieht; beide werden zur Kommunikation in NASA Projekten (Mars-Missionsn) und im Mobilfunk eingesetzt

Entwurf der Bayes'schen Netze

- Der Experte muss die kausalen Abhängigkeiten definieren und quantifizieren, d.h. er muss die Tafeln der bedingten Wahrscheinlichkeit auffüllen
- Gerade, wenn eine Variable viele Eltern besitzt kann dies schwierig bis unmöglich sein; man bedient sich gewisser vereinfachender Strukturen; am bekanntesten ist das Noisy-Or

Noisy-Or

- Sei $X \in \{0, 1\}$ eine binäre Variable mit Eltern $U_1, \ldots, U_n$
- Sei q_i die Wahrscheinlichkeit, dass $X = 0$, wenn $U_i = 1$ und alle anderen $U_j = 0$; dies bedeutet, dass $c_i = 1 - q_i$ den Einfluss einer einzelnen Größe bewertet;
- Dann ist $P(X = 0 | U_1 = 1, \ldots, U_n = 1) = \prod_{i=1}^{n} q_i$
- Dies bedeutet, dass Ereignisse immer wahrscheinlicher werden, wenn mehrere der Bedingungen erfüllt sind
- Beispiel: Wenn Krankheit A Fieber mit Wahrscheinlichkeit $c_A = 1 - q_A$ erzeugt, und Krankheit B Fieber mit Wahrscheinlichkeit $c_B = 1 - q_B$ erzeugt, dann ist die Wahrscheinlichkeit von Fieber wenn beide Krankheiten vorliegen gleich $1 - q_A q_B$. Dieser Ausdruck ist größer als c_A und c_B

Lernen in BNs

- Wir nehmen an, dass Daten vollständig sind
- ML-Parameterlernen ist in diesen Systemen denkbar einfach: man zählt einfach ab
- Betrachten wir den Parameter $\theta_{i,j,k}$, der definiert ist als

$$\theta_{i,j,k} = P(X_i = k | \text{par}(X_i) = j)$$

- Dies bedeutet, dass $\theta_{i,j,k}$ gleich der Wahrscheinlichkeit ist, dass X_i im Zustand k ist, wenn die Eltern in Zustand j sind (wir nehmen an, dass man die Zustände der Eltern in einer systematischen Art nummerieren kann).
- Dann ist die ML Schätzung

$$\theta_{i,j,k}^{ML} = \frac{N_{i,j,k}}{\sum_k N_{i,j,k}}$$

Hierbei ist $N_{i,j,k}$ die Anzahl der Trainingsdaten, bei denen $X_i = k, \text{par}(X_i) = j$.

EM-Lernen

- Haben die Trainingsdaten fehlende Einträge oder versteckte Variablen, kann man die ML-Lösung mit dem EM Algorithmus finden
- Im E-Schritt (*Expectation*) wird mit den Inferenzverfahren (Belief Propagation, Junction tree algorithm, ...) berechnet

$$E(N_{i,j,k}) = \sum_{l=1}^{N} P(X_i = k, \text{par}(X_i) = j | d_l, \theta)$$

  d_l ist der l-te möglicherweise unvollständige Datenpunkt und E steht für den Erwartungswert. Dies bedeutet, dass man im E-schritt die gemeinsame Verteilung eines Knotens und seiner Eltern berechnet, gegeben den aktuellen Datenpunkt und gegeben die aktuellen Parameterwerte θ

- Im M-Schritt (*Maximization*) benutzt man die Schätzungen

$$\theta_{i,j,k}^{ML} = \frac{E(N_{i,j,k})}{\sum_k E(N_{i,j,k})}$$

- E-schritt und M-schritt werden bis zur Konvergenz iteriert
- Der EM Algorithmus spielt allgemein (nicht nur bei Bayes Netzen) beim Lernen mit fehlenden Daten und latenten Variablen eine zentrale Rolle

EM-Lernen beim HMM und beim Kalman Filter

- HMMs werden mit dem Baum-Welch-Algorithmus trainiert; dies entspricht genau dem entsprechenden EM-Algorithmus
- Offline Versionen zum Training des Kalman Filters benutzen ebenfalls den EM-Algorithmus

Einbeziehen von Vorwissen

- Vorwissen kann leicht integriert werden
- Ich vergeben eine effektive Anzahl von Counts $\alpha_{i,j,k} \geq 0$, die meinem a priori Glauben entsprechen
- Dann ist

$$\theta^{ML}_{i,j,k} = \frac{\alpha_{i,j,k} + N_{i,j,k}}{\sum_k \alpha_{i,j,k} + N_{i,j,k}}$$

- Entsprechend im EM-Algorithmus

$$\theta^{ML}_{i,j,k} = \frac{\alpha_{i,j,k} + E(N_{i,j,k})}{\sum_k \alpha_{i,j,k} + E(N_{i,j,k})}$$

Strukturlernen in Bayes'schen Netzen

- Beim Strukturlernen muss man einige Dinge beachten
- So gibt es äquivalente Modelle, die sich basierend auf Daten nicht unterscheiden lassen, da sie die gleichen Unabhängigkeiten ausdrücken; A->B, B->A (independence equivalence)
- Man darf die Pfeilrichtungen nur unter sehr starken Annahmen kausal interpretieren; typische Annahmen, die man machen muss, ist dass die wahre Domäne kausal ist (kausales Weltmodell) und dass es keine wesentlichen Variablen gibt, die im Modell nicht enthalten sind (vollständiges Weltmodell); sogar dan können typischweise nicht alle kausalen Richtungen eindeutig identifiziert werden

Strukturlernen in Bayes'schen Netzen (2)

- Die populärsten Ansätze definieren eine Kostenfunktion, die man in der Struktursuche mit einer Suchstrategie optimiert
- Außer in trivial kleinen Netzen kann man nicht alle möglichen Strukturen testen
- Greedy Search: Man beginnt mit einem Startnetz (vollvernetzt, leer, ...), macht lokale Änderungen und akzeptiert eine Änderung, falls dadurch die Kostenfunktionsich verbessert wird
- Greedy search kann man mehrmals mit verschiedenen Startpunkten durchführen
- Alternativen: Simulated Annealing, Genetische Algorithmen

Kostenfunktionen

- Als Kostenfunktion kann man die Log-Likelihood auf einer Testmenge verwenden oder man wählt einen Kreuzvalisierungsansatz
- Beliebt is ebenfalls das BIC (Bayesian Information Criterion) Kriterium, maximiere:

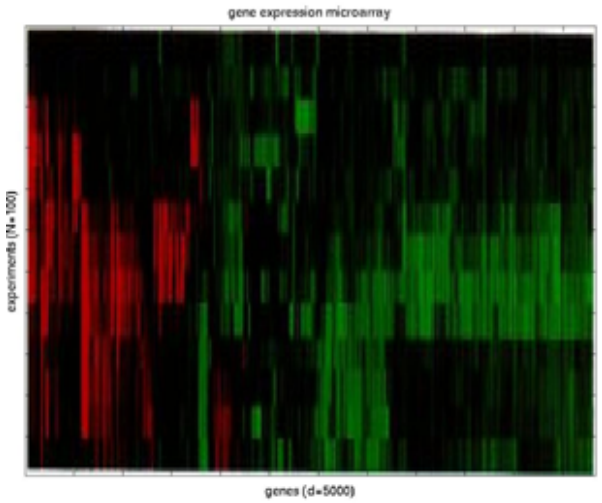
$$\log L - \frac{M}{2} \log N$$

(M ist die Anzahl der Parameter und N ist die Anzahl der Datenpunkte)

- Für komplette Daten ohne fehlende oder latente Variablen kann man explizit $P(D|M)$ berechnen und eine saubere Bayes'sche Modellselektion durchführen

Structure learning (data mining)

Gene expression data



Genetic pathway

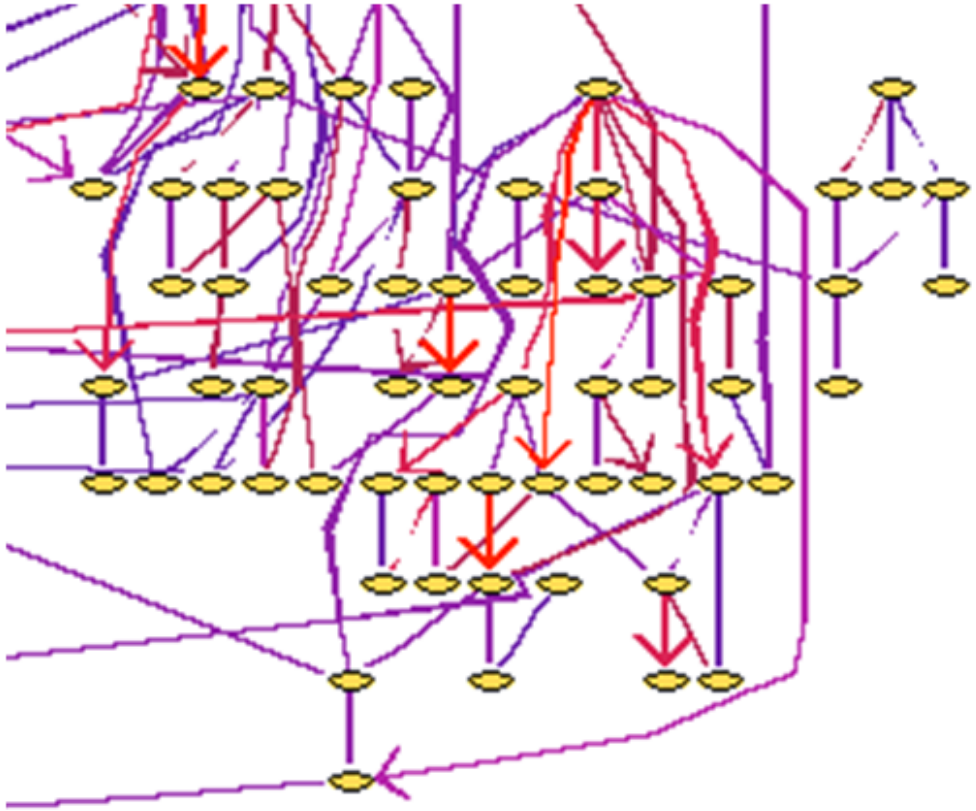


Figure from N. Friedman

Weitere Verfahren zur Strukturoptimierung

- Multiple-jump Markov Chain Monte Carlo (MCMC) Methoden; voll Bayes'scher Ansatz
- Verfahren, die direkt Unabhängigkeiten abtesten (*constraint-based methods*)

A comparison of GM software

Name	Authors	Src.	Cts	GUI	θ	G	Free
Analytica	Lumina	N	Y	W	N	N	N
Bayda	U. Helsinki	Java	Y	Y	Y	N	F
BayesBuilder	Nijman (Nijmegen)	N	N	Y	N	N	N
B. Knl. Disc.	KMI/Open U.	N	D	Y	Y	Y	F
B-course	U. Helsinki	N	D	Y	Y	Y	F
BN pow. cstr.	Cheng (U.Alberta)	N	N	Y	Y	Y	F
BN Toolbox	Murphy (UCB)	Matlab	Y	N	Y	Y	F
BucketElim	Rish (UCI)	C++	N	N	N	N	F
BUGS	MRC/Imperial	N	Y	W	Y	N	F

www.ai.mit.edu/~murphyk/Software/Bayes/bnsoft.html

CIspace	Poole (UBC)	Java	N	Y	N	N	F
Ergo	Noetic Systems	N	N	Y	N	N	N
Genie/Smile	U. Pittsburgh	N	N	W	N	N	F
Hugin Light	Hugin	N	Y	W	N	N	N
Ideal	Rockwell	Lisp	N	Y	N	N	F
Java Bayes	Cozman (CMU)	Java	N	Y	N	N	F
MIM	HyperGraph	N	Y	Y	Y	Y	N
MSBN	Microsoft	N	N	W	N	N	F
Netica	Norsys	N	Y	W	Y	N	N
Pronel	Hugin	N	N	W	Y	Y	F
RISO	Dodier (Colorado)	Java	Y	Y	N	N	F
Tetrad	CMU	N	Y	N	Y	Y	F

Bemerkungen

- Markov Netze sind ähnlich zu Bayes Netzen, allerdings sind hier die Kanten ungerichtet und Markov Netze besitzen keine kausale Interpretation
- Bayes'sche Netze und Markov Netze sind die wichtigsten Vertreter der Graphischen Modelle (*graphical models*)
- Neben diskreten Variablen gibt es Netze aus stetigen Gauss'schen Variablen und weitere Varianten ....