Inference in Bayesian Networks
How to generate random vectors from a Bayesian network?

- Sample parents first
  - \( P(C) \)
    - \((0.5, 0.5) \rightarrow yes\)
  - \( P(S|C=yes) \)
    - \((0.9, 0.1) \rightarrow on\)
  - \( P(R | C=yes) \)
    - \((0.8, 0.2) \rightarrow no\)
  - \( P(W | S=on, R=no) \)
    - \((0.9, 0.1) \rightarrow yes\)
  - \( P(C,S,R,W) = P(yes,on,no,yes) = 0.5 \times 0.9 \times 0.2 \times 0.9 = 0.081 \)
Two types of probabilistic reasoning

- \( n \) (discrete) random variables \( X_1, \ldots, X_n \)
- joint probability distribution \( P(X_1, \ldots, X_n) \)
- Input: a partial value assignment \( \Omega \),
  \( \Omega = < X_1, X_2=x_2, X_3, X_4=x_4, X_5=x_5, X_6, \ldots, X_n > \)

  - **Probabilistic reasoning, type I (marginal distribution):**
    - compute \( P(X=x| \Omega) \) for some \( X \) not instantiated in \( \Omega \),
      and for all values \( x \) of \( X \).

  - **Probabilistic reasoning, type II (MAP assignment):**
    - Given \( \Omega \), find a maximum a posterior probability value
      assignment jointly for all the \( X_i \) not instantiated in \( \Omega \)

- N.B. These are not the same thing!

- Bayesian networks: a family of probabilistic models and
  algorithms enabling computationally efficient probabilistic reasoning
Some famous (simple) Bayesian network models

- Naïve Bayes classifier
- Finite mixture model
- Tree Augmented Naïve Bayes
- Hidden Markov Models (HMMs)
Naïve Bayes classifier

\[ P(C) \]

\[ \text{Class} \]

\[ P(X_1|C) \rightarrow X_1 \]
\[ P(X_2|C) \rightarrow X_2 \]
\[ P(X_3|C) \rightarrow X_3 \]
\[ P(X_4|C) \rightarrow X_4 \]
\[ P(X_5|C) \rightarrow X_5 \]

\[ X_i \text{ are called predictors or indicators} \]
Naïve Bayes Classifier

- Structure tailored for efficient diagnostics $P(C|x_1, x_2, \ldots, x_n)$.
  - Obs! Does NOT try to model directly the target probability distribution $P(C|x_1, x_2, \ldots, x_n)$

- Unrealistic conditional independence assumptions, but OK for the particular query $P(C|x_1, x_2, \ldots, x_n)$.

- Because of wrong independence assumptions, NB is often poorly calibrated:
  - Probabilities $P(C|x_1, x_2, \ldots, x_n)$ may be way off, but $\arg\max_c P(c|x_1, x_2, \ldots, x_n)$ still often correct.
Calculating $P(C|x_1, x_2, \ldots, x_n, \text{NB})$

- Boldly calculate through joint probability

$$P(C|x_1, \ldots, x_n) \propto P(C, x_1, \ldots, x_n) = P(C) \prod_{i=1}^{n} P(x_i|C)$$

- No need to have all the predictors. Having just set $X_A$ of predictors (and not $X_B$):

$$P(C|x_A) \propto P(C, x_A) = \sum_{x_B} P(C, x_A, x_B)$$

$$= \sum_{x_B} P(C) \prod_{i \in A} P(x_i|C) \prod_{j \in B} P(x_j|C)$$

$$= P(C) \prod_{i \in A} P(x_i|C) \sum_{x_B} \prod_{j \in B} P(x_j|C)$$

$$= P(C) \prod_{i \in A} P(x_i|C) \prod_{j \in B} \sum_{x_j} P(x_j|C) = P(C) \prod_{i \in A} P(x_i|C)$$
Example

6 binary variables: C, X₁,...X₅, \( P(C=0)=0.4 \)

\[
P(C=0 \mid X₁=0,X₂=1,X₃=0,X₄=1,X₅=0)
\]
\[
\alpha 0.4 \times 0.8 \times 0.5 \times 0.4 \times 0.3 \times 0.9 = 0.017 \quad 17/27=63\%
\]

\[
P(C=1 \mid X₁=0,X₂=1,X₃=0,X₄=1,X₅=0)
\]
\[
\alpha 0.6 \times 0.2 \times 0.3 \times 0.6 \times 0.8 \times 0.6 = 0.010 \quad 10/27=37\%
\]

\[
P(C=0 \mid X₂=1,X₃=0,X₄=1,X₅=0)
\]
\[
\alpha 0.4 \times 0.5 \times 0.4 \times 0.3 \times 0.9 = 0.022 \quad 22/74=30\%
\]

\[
P(C=1 \mid X₂=1,X₃=0,X₄=1,X₅=0)
\]
\[
\alpha 0.6 \times 0.3 \times 0.6 \times 0.8 \times 0.6 = 0.052 \quad 52/74=70\%
\]
Tree Augmented Naïve Bayes (TAN)

- $X_i$ may have at most one other $X_j$ as an extra parent.
Calculating $P(C|x_1, x_2, \ldots, x_n, \text{TAN})$

- Again, boldly calculate via joint probability

$$P(C|x_1, \ldots, x_n) \propto P(C, x_1, \ldots, x_n) = P(C) \prod_{i=1}^{n} P(x_i|C, Pa(x_i))$$

- But missing predictors may hurt more. For example:

$$P(C|x_5) \propto P(C) P(x_5|C) = P(C) \sum_{x_4} P(x_4|C) P(x_5|x_4, C)$$
$$= P(C) \sum_{x_4} P(x_5|C, x_4) P(x_4|C)$$
$$= P(C) \sum_{x_4} P(x_5|C, x_4) \sum_{x_3} P(x_4|C, x_3) P(x_3|C)$$
$$= \ldots$$
NB as a Finite Mixture Model

- When the Naive Bayes structure is reasonable, it also makes a nice (marginal) joint probability model $P(X_1, X_2, \ldots, X_n)$ for “predictors”.

- A computationally effective alternative for building a Bayesian network for $X_1, X_2, \ldots, X_n$.

- Joint probability $P(X_1, X_2, \ldots, X_n)$ is represented as a mixture of $K$ joint probability distributions $P_k(X_1, X_2, \ldots, X_n) = P_k(X_1)P_k(X_2)\ldots P_k(X_n)$, where $P_k(\cdot) = P(\cdot | C=k)$. 
Calculating with $P(X_1, X_2, \ldots, X_n | NB)$

- Joint probability a simple marginalization:

$$P(X_1, \ldots, X_n) = \sum_{k=1}^{K} P(X_1, \ldots, X_n, C=k)$$

$$= \sum_{k=1}^{K} P(C=k) \prod_{i=1}^{n} P(X_i | C=k)$$

- Inference

$$P(X | e) \propto P(e, X) = \sum_{k=1}^{K} P(e, X, C=k)$$

$$= \sum_{k=1}^{K} P(C=k) P(e, X | C=k)$$

$$= \sum_{k=1}^{K} P(C=k) \prod_{X_i \in X} P(X_i | C=k) \prod_{e_i \in e} P(e_i | C=k)$$
Example

- Consider the previous example (the NB model).
- What is $P(X_4 | X_5=0)$?

  - $P(X_4=0, X_5=0 | C=0) = 0.7 \times 0.9 = 0.63$
  - $P(X_4=1, X_5=0 | C=0) = 0.3 \times 0.9 = 0.27$
  - $P(X_4=0, X_5=0 | C=1) = 0.2 \times 0.6 = 0.12$
  - $P(X_4=1, X_5=0 | C=1) = 0.8 \times 0.6 = 0.48$
  - $P(X_4=0, X_5=0) = P(X_4=0, X_5=0 | C=0)P(C=0) + P(X_4=0, X_5=0 | C=1)P(C=1) = 0.63 \times 0.4 + 0.12 \times 0.6 = 0.324$
  - $P(X_4=1, X_5=0) = P(X_4=1, X_5=0 | C=0)P(C=0) + P(X_4=1, X_5=0 | C=1)P(C=1) = 0.27 \times 0.4 + 0.48 \times 0.6 = 0.396$
  - $P(X_4=0 | X_5=0) = P(X_4=0,X_5=0)/P(X_5=0) = 0.45$
  - $P(X_4=1 | X_5=0) = P(X_4=1,X_5=0)/P(X_5=0) = 0.55$
Hidden Markov Models

- Temporal/sequential probabilistic models
- States of the process are hidden but an output dependent on the hidden state is observable
- Frequently applied in e.g. speech recognition, robot navigation, and other pattern recognition tasks
Markov chains

- Assume that the world has a finite number of states, and the changes in the world are caused by a stationary process:
  - The process does not change over time
- The world has a Markov property:
  - The current state depends only on a finite history of previous states
- A Markov chain is a sequence of random variables \(X_0, X_1, X_1, \ldots\) with the Markov property
  - We mainly consider first-order Markov chains where \(P(X_t | X_{0:t-1}) = P(X_t | X_{t-1})\)
Hidden Markov Models

- Models observations about a system that changes its state.

\[ P(X_{t+1} | X_t) \]

\[ P(e_t | X_t) \]

\[ P(e_{t+1} | X_{t+1}) \]

Transition model

Sensor model

No colliding arcs, thus independences are easy to determine.

NB! Sensor model does not depend on time \( t \).
Hidden Markov Model as a BN

- For inference, easier to think of as a long chain of variables
- (For learning, the two-state model more fitting)
- No head-to-head nodes!
- Node $X_t$ represents the (hidden) state at time $t$, and $E_t$ is the observation at time $t$
Hidden Markov Model as a BN

- For inference, easier to think of as a long chain of variables
- (For learning, the two-state model more fitting)
- No head-to-head nodes!
- Node $X_t$ represents the (hidden) state at time $t$, and $E_t$ is the observation at time $t$
Graphical models on Manhattan — A probabilistic approach to mobile device positioning
Location positioning problem
The positioning problem

- Given some location-dependent observations $O$, measured by a mobile device, determine the location $L$ of the device.

- Why is this a good research problem?
  - The goodness of different solutions is extremely easy to validate (just go to a known location and test).
  - The results have immediate practical applications:
    - Location-based services (LBS)
    - FCC Enhanced 911:
      - Network-based solutions: error below 100 meters for 67 percent of calls, 300 meters for 95 percent of calls
      - Handset-based solutions: error below 50 meters for 67 percent of calls, 150 meters for 95 percent of calls
Cell ID

variable cell-size:
n. 50 m (indoors) --> 30 km (rural areas)
Cell-id in urban positioning

- errors > 500m common
+ simple
Cell ID errors
Enhanced Observed Time Difference (E-OTD)
Problems with E-OTD in urban positioning

- multi-paths
- no line of sight to BS
- extra hardware
"Theory"
The signal propagation approach

Theory

Reality
Empirical modeling in urban positioning

+ accurate
+ handset or network based
- calibration measurements required
A probabilistic approach to positioning

Bayes rule: \[ P(L \mid O) = \frac{P(O \mid L) P(L)}{P(O)} \]

- A probabilistic model assigns a probability for each possible location \( L \) given the observations \( O \).
  - \( P(O \mid L) \) is the conditional probability of obtaining observations \( O \) at location \( L \).
  - \( P(L) \) is the prior probability of location \( L \). (Could be used to exploit user profiles, rails etc.)
  - \( P(O) \) is just a normalizing constant.
- How to obtain \( P(O \mid L) \)? \( \Rightarrow \) Empirical observations + machine learning
Tracking with Markov models

- Typically we have a sequence (history) of observations $O_1, \ldots, O_n$, and wish to determine $P(L_n | O^n)$.
- Assumption: $P(O_t | L_t)$ are known, and given location $L_t$, the observation $O_t$ is independent of the rest of the history.
- The model: a hidden Markov model (HMM) where the locations $L_t$ are the hidden unobserved states.
- The transition probabilities $P(L_t | L_{t-1})$ can be easily determined from the physical properties of the moving object.

Diagram:

```
L_1 -----> L_2 -----> ... -----> L_{n-1} -----> L_n
|               |               |               |               |
|               |               |               |               |
|               |               |               |               |
O_1                                 O_{n-1} -----> O_n
```

$O_1, \ldots, O_n$, are the observations.
One more assumption

- The observation at time $t$ typically consists of several measurements (e.g., strengths of signals from all the transmitters that can be heard)

- If the wireless network is designed in a reasonable manner (the transmitters are far from each other), it makes sense to assume that the individual observations are independent, given the location

- The “Naïve Bayes” model
The Model

First-order "semi-hidden" Markov model
Tracking as probabilistic inference

• As our hidden Markov model is a tree, we can compute the marginal of any $L_t$, given the history $O^n$, in linear time by using a simple forward-backward algorithm.

• Alternatively, we can compute the maximum probability path $L_1, \ldots, L_n$ given the history (this is known as the Viterbi algorithm).

• **Kalman filter**: all the conditional distributions of the HMM model are normal distributions (linear dependencies with Gaussian noise).
Recursive tracking

• Assume that $P(L_{n-1} \mid O^{n-1})$ has been computed.

• Our model defines the transition probabilities $P(L_t \mid L_{t-1})$ and the local observation probabilities $P(O_t \mid L_t)$

• Now $P(L_n \mid O^n) \propto P(L_n, O^n)$
  
  
  $$= P(O_n \mid L_n, O^{n-1}) P(L_n, O^{n-1})$$

  
  $$= P(O_n \mid L_n) \sum_{L_{n-1}} P(L_n, L_{n-1}, O^{n-1})$$

  
  $$\propto P(O_n \mid L_n) \sum_{L_{n-1}} P(L_n \mid L_{n-1}) P(L_{n-1} \mid O^{n-1})$$

• With a Kalman filter, the recursive process operates all the time with Gaussians
GSM-positioning trials
NYC Trial 2001

http://cosco.hiit.fi/demo/manhattan/
Details

- Covering downtown Manhattan (10th - 114th St)
- Data gathering by car
- Modeling: 10 person days
- Target accuracy: less than 911 handset requirements
- Tests using cars
Accuracy of NYC Trial 2001

- 20166 points
- tracking; testing done in a car;
Trials: Manhattan 2002

GSM locationing demo on Manhattan Times Square
Challenges

• “real 911” simulation
  - No tracking information
  - Only up to 60 seconds of signal measurements
• Target accuracy: “theater level”
• Indoor testing (without indoor modeling)
Accuracy NYC Trial 2002

- 30 points
- static; testing done by walking;
WiFi-positioning


More information: www.ekahau.com
Thesis topic: semi-supervised modeling in positioning

- "automatic calibration"
Joint probability of a HMM

- Joint probability factorizes like a BN
  - HMM is a Bayesian network!

\[
P(X_0, X_1, E_1, X_2, E_2, \ldots, X_t, E_t) = P(X_0) \prod_{i=1}^{t} P(X_i | X_{i-1}) P(E_i | X_i)
\]

- Common inference tasks:
  - Filtering / monitoring: \( P(X_t | e_{1:t}) \)
  - Prediction: \( P(X_{t+k} | e_{1:t}), k>0 \)
  - Smoothing: \( P(X_k | e_{1:t}), k<t \)
  - Explanation: \( \arg \max_{x_{1:t}} P(X_{1:t} | e_{1:t}) \)
Inference tasks visualized

**Filtering**

\[
\begin{align*}
X_1 & \rightarrow X_2 & \rightarrow X_k & \rightarrow X_{t-1} & \rightarrow X_t \\
E_1 & \rightarrow E_2 & \rightarrow E_k & \rightarrow E_{t-1} & \rightarrow E_t
\end{align*}
\]

**Prediction**

\[
\begin{align*}
X_1 & \rightarrow X_2 & \rightarrow X_k & \rightarrow X_{t-1} & \rightarrow X_t \\
E_1 & \rightarrow E_2 & \rightarrow E_k & \rightarrow E_{t-1} & \rightarrow E_t
\end{align*}
\]

**Smoothing**

\[
\begin{align*}
X_1 & \rightarrow X_2 & \rightarrow X_k & \rightarrow X_{t-1} & \rightarrow X_t \\
E_1 & \rightarrow E_2 & \rightarrow E_k & \rightarrow E_{t-1} & \rightarrow E_t
\end{align*}
\]

**Most likely sequence**

\[
\begin{align*}
X_1 & \rightarrow X_2 & \rightarrow X_k & \rightarrow X_{t-1} & \rightarrow X_t \\
E_1 & \rightarrow E_2 & \rightarrow E_k & \rightarrow E_{t-1} & \rightarrow E_t
\end{align*}
\]
Calculating $P(X_t | e_{1:t})$ in HMM

- Let's shoot for a recursive formula:

$$P(X_{t+1} | e_{1:t+1}) = P(X_{t+1} | e_{t+1}, e_{1:t})$$

$$\propto P(e_{t+1} | X_{t+1}, e_{1:t}) P(X_{t+1} | e_{1:t})$$

$$= P(e_{t+1} | X_{t+1}) P(X_{t+1} | e_{1:t})$$

- and

$$P(X_{t+1} | e_{1:t}) = \sum_{x_t} P(X_{t+1}, x_t | e_{1:t})$$

$$= \sum_{x_t} P(X_{t+1} | x_t, e_{1:t}) P(X_t | e_{1:t})$$

$$= \sum_{x_t} P(X_{t+1} | x_t) P(x_t | e_{1:t})$$
Forward algorithm for $P(X_t | e_{1:t})$

- Combining formulas we get a recursion

$$P(X_{t+1}|e_{1:t+1}) \propto P(e_{t+1}|X_{t+1}) \sum_{x_t} P(X_{t+1}|x_t) P(x_t|e_{1:t})$$

- So first calculate

$$P(X_1|e_1) \propto P(e_1|X_1) \sum_{x_0} P(X_1|x_0) P(x_0)$$

- and then

$$P(X_2|e_1,e_2) \propto P(e_2|X_2) \sum_{x_1} P(X_2|x_1) P(x_1|e_1)$$

$$P(X_3|e_1,e_2,e_3) \propto P(e_3|X_3) \sum_{x_2} P(X_3|x_2) P(x_2|e_1,e_2)$$
Prediction: $P(X_{t+k} | e_{1:t})$, $k > 0$

- $P(X_{t+1} | e_{1:t})$ part of the forward algorithm

- and from that on evidence does not count, and one can just calculate forward:

\[
P(X_{t+2} | e_{1:t}) = \sum_{x_{t+1}} P(X_{t+2} | x_{t+1}, e_{1:t}) P(x_{t+1} | e_{1:t}) \\
= \sum_{x_{t+1}} P(X_{t+2} | x_{t+1}) P(x_{t+1} | e_{1:t}) \\

P(X_{t+3} | e_{1:t}) = \sum_{x_{t+2}} P(X_{t+3} | x_{t+2}, e_{1:t}) P(x_{t+2} | e_{1:t}) \\
= \sum_{x_{t+2}} P(X_{t+3} | x_{t+2}) P(x_{t+2} | e_{1:t})
\]
Smoothing: \( P(X_k \mid e_1:t), \ k < t \)

- **Obvious move**: divide \( e_1:t \) to \( e_1:k \) and \( e_{k+1:t} \).

\[
P(X_k \mid e_1:t) = P(X_k \mid e_1:k, e_{k+1:t}) \propto P(X_k \mid e_1:k) P(e_{k+1:t} \mid X_k, e_1:k)
\]

\[
P(e_{k+1:t} \mid X_k) = \sum_{x_{k+1}} P(x_{k+1}, e_{k+1:t} \mid X_k) = \sum_{x_{k+1}} P(x_{k+1} \mid X_k) P(e_{k+1:t} \mid x_{k+1}, X_k) = \sum_{x_{k+1}} P(x_{k+1} \mid X_k) P(e_{k+1} \mid e_{k+2:t} \mid x_{k+1}) = \sum_{x_{k+1}} P(x_{k+1} \mid X_k) P(e_{k+1} \mid x_{k+1}) P(e_{k+2:t} \mid x_{k+1})
\]

- **and the first (last) step**:

\[
P(e_t \mid X_{t-1}) = \sum_{x_t} P(x_t, e_t \mid X_{t-1}) = \sum_{x_t} P(e_t \mid x_t, X_{t-1}) P(x_t \mid X_{t-1}) = \sum_{x_t} P(e_t \mid x_t) P(x_t \mid X_{t-1})
\]
Back and forth

- "Brute-force" smoothing of the whole sequence takes $O(t^2)$ time
- *Forward-backward* algorithm: $O(t)$
- Finding the most probable sequence works in the same manner (the Viterbi algorithm / Viterbi path)
Finding the most probable sequence

• Want to compute:

\[
\max_{X_1, \ldots, X_n} P(X_1, \ldots, X_n | e_1, \ldots, e_n) = \max_{X_n} \max_{X_1, \ldots, X_{n-1}} P(X_1, \ldots, X_{n-1}, X_n, e_1, \ldots, e_n)
\]

• Recursion:

\[
\max_{X_1, \ldots, X_{n-1}} P(X_1, \ldots, X_{n-1}, X_n | e_1, \ldots, e_n) = \max_{X_1, \ldots, X_{n-1}} P(X_1, \ldots, X_{n-1}, X_n, e_1, \ldots, e_n)
\]
\[
= \max_{X_1, \ldots, X_{n-1}} P(e_n | X_n, X_1, \ldots, X_{n-1}, e_1, \ldots, e_{n-1}) P(X_n, X_1, \ldots, X_{n-1}, e_1, \ldots, e_{n-1})
\]
\[
= \max_{X_1, \ldots, X_{n-1}} P(e_n | X_n) P(X_n | X_1, \ldots, X_{n-1}, e_1, \ldots, e_{n-1}) P(X_1, \ldots, X_{n-1}, e_1, \ldots, e_{n-1})
\]
\[
= P(e_n | X_n) \max_{X_{n-1}} P(X_n | X_{n-1}) \max_{X_1, \ldots, X_{n-2}} P(X_1, \ldots, X_{n-2}, X_{n-1} | e_1, \ldots, e_{n-1})
\]

• More:

  - see e.g. Russel & Norvig, Chapter 15.2.
The Viterbi algorithm

Let 
\[ p(X, i) = \max_{X_1, \ldots, X_{i-1}} P(X_1, \ldots, X_{i-1}, X | e_1, \ldots, e_i) \]
denote the probability of the most probable sequence of length \( i \) ending in state \( X \).

\[
p(X, 1) = P(e_1 | X) P(X) = P(e_1 | X) \sum_{X_0} (P(X | X_0) P(X_0))
\]
\[
p(X, i) = P(e_i | X) \max_Y [p(Y, i-1) P(X | Y)], \text{ for } i > 1.
\]
The Viterbi algorithm

Let 

\[ p(X, i) = \max_{X_1, \ldots, X_{i-1}} P(X_1, \ldots, X_{i-1}, X | e_1, \ldots, e_i) \]

denote the probability of the most probable sequence of length \( i \) ending in state \( X \).

\[ p(X, 1) = P(e_1|X)P(X) = P(e_1|X) \sum_{X_0} (P(X|X_0)P(X_0)) \]

\[ p(X, i) = P(e_i|X) \max_{Y} [p(Y, i-1)P(X|Y)], \text{ for } i > 1. \]
Probabilistic inference in DAGs
Types of inference

- Assume that both the structure of the model (the DAG), and the parameters (local probability tables) are fixed.

- Recall the two types of inference task: either compute the conditional probability of a (set of) variables, given the values of others, or compute the maximum probability assignment.

- Inference can be either exact or approximative.
Exact inference in singly-connected BNs

- a singly connected BN = polytree (disregarding the arc directions, no two nodes can be connected with more than one path).
Probabilistic reasoning in singly-connected BNs

\[ P(X|E) \propto P(X, E_+, E_-) \propto P(E_-|X)P(X|E_+) \]
\[ P(E_-|X) = \prod_{Y} P(E_{Y-}|X) \]
\[ P(E_{Y-}|X) = \sum_{Y} P(E_{Y-}|Y)P(Y|X) \]
\[ P(X|E_+) = \sum_{Z} P(X|Z)P(Z|E_{Z+}) \]

- a computationally efficient **message-passing** scheme: time requirement linear in the number of conditional probabilities in \( \Theta \).
Belief propagation

- A message passing algorithm developed by Judea Pearl
- Computes the marginal distribution of an unobserved variable given the observed ones
- Each node maintains a belief of its state (the conditional probability distribution, given the evidence)
- Nodes pass messages to their neighbors and update their beliefs based on received messages
Belief propagation in chains

- A node can have at most one parent and child, no loops.
- We want to compute the marginal probability $P(X \mid e)$, where the evidence $e$ is an instantiation of node set $E$.
- Let us partition the evidence $e$ into evidence from “upstream” $e^+$ and evidence from “downstream” $e^-$.

\[
P(X \mid e) = P(X \mid e^+, e^-) \\
\propto P(e^- \mid X, e^+)P(X \mid e^+) \\
= P(e^- \mid X)P(X \mid e^+)
\]
Message passing in chains

\[ \lambda(U = u) = P(e^- | U = u) \]
\[ = \sum_x P(e^- | X = x)P(X = x | U = u) \]
\[ = \sum_x \lambda(X = x)P(X = x | U = u) \]

\[ \pi(X = x) = P(X = x | e^+) \]
\[ = \sum_u P(X = x | U = u)P(U = u | e^+) \]
\[ = \sum_u P(X = x | U = u)\pi(U = u) \]
Initialization

• For nodes $E$ with evidence $e$:

\[
\lambda(E = e) = 1, \text{otherwise } \lambda(E = x) = 0
\]

\[
\pi(E = e) = 1, \text{otherwise } \pi(E = x) = 0
\]

• Nodes with no parents:

\[
\pi(x) = P(x) \quad (\text{prior probabilities})
\]

• Nodes with no children:

\[
\lambda(x) = 1, \text{for all } x
\]
Belief propagation in trees

- Every node has at most one parent.
- Differences compared to chains:
  - Each node must combine impacts of the $\lambda$-messages obtained from its children.
  - Each node should distribute a separate $\pi$-message to each of its children.
Message passing in trees

Initialization like with chains. Then (in any order):

- **Belief updating:**
  \[
  \text{BEL}(x) = P(x|e) \propto \lambda(x) \pi(x).
  \]
  \[
  \lambda(x) = \prod_j \lambda_{Y_j}(x).
  \]
  \[
  \pi(x) = \sum_u P(x|u) \pi_X(u).
  \]

- **Bottom-up propagation:**
  \[
  \lambda_X(u) = \sum_x \lambda(x) P(x|u).
  \]

- **Top-down propagation:**
  \[
  \pi_{Y_j}(x) \propto \pi(x) \prod_{k \neq j} \lambda_{Y_k}(x).
  \]
Belief propagation in polytrees

- Nodes can have multiple parents
- No loops
- Differences compared to trees:
  - Each node must combine impacts of the $\pi$-messages obtained from its parents.
  - Each node should distribute a separate $\lambda$-message to each of its parents.
Message passing in polytrees

- For details, see e.g. Neapolitan (Chapter 3.2.), or Pearl (Chapter 4.2.)
Complexity

- Number of messages sent depends linearly on the diameter of the network
- The time needed to compute a message is linear with respect to the size of the local probability table
  - But note that this means that the time (and size) is exponential with respect to the number of parents!
- The message-passing algorithm does not work with multi-connected networks
Probabilistic reasoning in multi-connected BNs

- Generally not computationally feasible as the problem has been shown to be NP-hard (Cooper 1990, Shimony 1994).

- Exact methods:
  - clustering
  - conditioning
  - variable elimination

- Approximative methods:
  - stochastic sampling algorithms
  - loopy belief propagation

- Even approximative inference (both in terms of absolute and relative error) is NP-hard
Variable elimination

- Idea: eliminate (marginalize) one variable at a time
- Usually, each step depends on a limited number of variables only
- Time (and space) complexity of the algorithm depends on the structure of the network, and on the elimination order
Variable elimination: a simple example

\[ P(D) = \sum_{A,B,C} P(A,B,C,D) \]

\[ = \sum_C \sum_B \sum_A P(A) P(B|A) P(C|B) P(D|C) \]

\[ = \sum_C \sum_B P(C|B) P(D|C) \sum_A P(A) P(B|A) \]

\[ = \sum_C P(D|C) \sum_B P(C|B) \sum_A P(A) P(B|A) \]
Approximate inference in Bayesian networks

- How to estimate how probably it rains next day, if the previous night temperature is above the month average?
  - count rainy and non rainy days after warm nights (and count relative frequencies).

- Rejection sampling for $P(X|e)$:
  1. Generate random vectors $(x_r, e_r, y_r)$.
  2. Discard those that do not match $e$.
  3. Count frequencies of different $x_r$ and normalize.
Rejection sampling, bad news

• Good news first:
  - super easy to implement

• Bad news:
  - if evidence $\mathbf{e}$ is improbable, generated random vectors seldom conform with $\mathbf{e}$, thus it takes a long time before we get a good estimate $P(\mathbf{X}|\mathbf{e})$.
  - With long $\mathbf{E}$, all $\mathbf{e}$ are improbable.

• So called likelihood weighting can alleviate the problem a little bit, but not enough.
Gibbs sampling

- A **Markov Chain Monte Carlo (MCMC) method** that approximates the probability distribution by sampling from a "cleverly" selected Markov Chain.

- Given a Bayesian network for n variables $\mathbf{X} \cup \mathbf{E} \cup \mathbf{Y}$, calculate $P(\mathbf{X}|\mathbf{e})$ as follows:

  \[ N = \text{(associative) array of zeros} \]

  Generate random vector $\mathbf{x}, \mathbf{y}$.

  While not enough samples:

  for $V$ in $\mathbf{X}, \mathbf{Y}$:

  generate $v$ from $P(V \mid \text{MarkovBlanket}(V))$

  replace $v$ in $\mathbf{x}, \mathbf{y}$.

  $N[\mathbf{x}] += 1$

  print normalize($N[\mathbf{x}]$)
Sampling from the Markov blanket

\[
P(X | mb(X)) \\
= P(X | mb(x), \text{Rest}) \\
= \frac{P(X, mb(X), \text{Rest})}{P(mb(X), \text{Rest})} \\
\propto P(\text{All}) \\
= \prod_{X_i \in X} P(X_i | Pa(X_i)) \\
= P(X | Pa(X)) \prod_{C \in \text{ch}(X)} P(C | Pa(C)) \prod_{R \notin \{X \cup \text{ch}(X)\}} P(R | Pa(R)) \\
\propto P(X | Pa(X)) \prod_{C \in \text{ch}(X)} P(C | Pa(C))
\]
Why does it work

- All decent Markov Chains have a unique stationary distribution $P^*$ that can be estimated by simulation.
- Detailed balance of transition function $q$ and state distribution $P^*$ implies stationarity of $P^*$.
- Proposed $q = P(V|mb(V))$, and $P(X|e)$ form a detailed balance, thus $P(X|e)$ is a stationary distribution, so it can be estimated by simulation.
Markov Chains: stationary distribution

- Defined by transition probabilities $q(x \rightarrow x')$ between states, where $x$ and $x'$ belong to a set of states $X$.

- Distribution $P^*$ over $X$ is called stationary distribution for the Markov Chain $q$, if $P^*(x') = \sum_x P^*(x)q(x \rightarrow x')$.

- $P^*(X)$ can be found out by simulating Markov Chain $q$ starting from a random state $x_r$. 
Markov Chains: detailed balance

- Distribution P over X and a state transition distribution q are said to form a detailed balance, if for any states x and x',
  \[ P(x)q(x \rightarrow x') = P(x')q(x' \rightarrow x), \]
i.e. it is equally probable to witness transition from x to x' as it is to witness transition from x' to x.

- If P and q form a detailed balance,
  \[ \sum_x P(x)q(x \rightarrow x') = \sum_x P(x')q(x' \rightarrow x) = P(x')\sum_x q(x' \rightarrow x) = P(x'), \]
thus P is stationary.
Gibbs sampler as Markov Chain

- Consider $Z=(X,Y)$ to be states of a Markov chain, and $q((v,z_{\neg V})\rightarrow(v',z_{\neg V}))=P(v'|z_{\neg V},e)$, where $Z_{\neg V}=Z\{-V\}$. Now $P^*(Z)=P(Z|e)$ and $q$ form a detailed balance, thus $P^*$ is a stationary distribution of $q$ and it can be found with the sampling algorithm.

\[
P^*(z)q(z\rightarrow z') = P(z|e)P(v'|z_{\neg V}, e) = P(v,z_{\neg V}|e)P(v'|z_{\neg V}, e) = P(v|z_{\neg V}, e)P(z_{\neg V}|e)P(v'|z_{\neg V}, e) = P(v|z_{\neg V}, e)P(v', z_{\neg V}|e) = q(z'|z)P^*(z'), \text{ thus balance.}
\]
Loopy belief propagation

- What happens if you just keep iterating the message passing algorithm in a multi-connected network?
  - In some cases it produces the right results, or at least a good approximation
- Turbo codes
Welcome to B-Course

B-Course is a web-based interactive tutorial on Bayesian modeling, in particular dependence modeling. However, it is more than just a tutorial. It is also a free data analysis tool that makes it possible for you to use your own data as example data for the tutorial. Consequently B-Course can be used as an analysis tool for any research where dependence modeling based on data is of interest. B-Course can be freely used for educational and research purposes only. (Disclaimer)

B-Course facilities

B-Course will guide you through the trail of dependency modeling. You will learn about Bayesian modeling and inference using your own data as an example. In case you do not (yet) have any data sets to analyze, you can take a look on a model we have prepared, or you can select among public data sets provided in B-Course material and use the selected data as your example.

Along the trail you will find references to the B-Course library for more detailed information. We advise you to study those texts, because they are vital for truly understanding what is going on in the analysis. When you familiarize yourself with the background information, you can use B-Course as any other software tool to help you in the analysis of your data. If you publish the results, we as the designers of B-Course would appreciate that you acknowledge that the results were obtained by using B-Course.

» Read about the goals of B-Course

So let us play….