Inference in Bayesian Networks
Inference in Bayesian networks

- Given a Bayesian network B (i.e., DAG and CPTs), calculate \( P(\mathbf{X}|\mathbf{e}) \) where \( \mathbf{X} \) is a set of query variables and \( \mathbf{e} \) is an instantiation of observed variables \( \mathbf{E} \) (\( \mathbf{X} \) and \( \mathbf{E} \) separate).

- There is always the way through marginals:
  - normalize \( P(\mathbf{x},\mathbf{e}) = \sum_{y \in \text{dom}(\mathbf{Y})} P(\mathbf{x}, y, \mathbf{e}) \), where \( \text{dom}(\mathbf{Y}) \), is a set of all possible instantiations of the unobserved non-query variables \( \mathbf{Y} \).

- There are much smarter algorithms too, but in general the problem is NP hard (more later).
How to generate random vectors from a Bayesian network

- Sample parents first
  - $P(C)$
    - $(0.5, 0.5) \rightarrow \text{yes}$
  - $P(S|C=\text{yes})$
    - $(0.9, 0.1) \rightarrow \text{on}$
  - $P(R | C=\text{yes})$
    - $(0.8, 0.2) \rightarrow \text{no}$
  - $P(W | S=\text{on}, R=\text{no})$
    - $(0.9, 0.1) \rightarrow \text{yes}$
  - $P(C,S,R,W) = P(\text{yes, on, no, yes})$
    - $= 0.5 \times 0.9 \times 0.2 \times 0.9 = 0.081$
Some famous (simple) Bayesian network models

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

- $X_i$ are called predictors or indicators
Naïve Bayes Classifier

- Structure tailored for efficient diagnostics $P(C|x_1, x_2, ..., x_n)$.

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

- Because of wrong independence assumptions, NB is often poorly calibrated:
  - Probabilities $P(C|x_1, x_2, ..., x_n)$ way off, but $\arg\max_c P(c|x_1, x_2, ..., x_n)$ 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 $X_B$):

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

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

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

$$= P(C) \prod_{i \in A} P(x_i|C) \sum_{j \in B} \sum_{x_j} P(x_j|C) = P(C) \prod_{i \in A} P(x_i|C)$$
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 NB structure is right, 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 $P(X_1, X_2, \ldots, X_n | \text{NB})$

- 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

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

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

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

NB! Sensor model does not depend on time \( t \).

No colliding arcs, thus independences are easy to determine.
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) \(\rightarrow\) 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"
"Practice"
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:

\[
P(L | O) = \frac{P(O | L) P(L)}{P(O)}
\]

- A probabilistic model assigns a probability for each possible location \( L \) given the observations \( O \).
  - \( P(O | L) \) is the conditional probability of obtaining observations \( O \) at location \( L \).
  - \( P(L) \) is the prior probability of location \( O \). (Could be used to exploit user profiles, rails etc.)
  - \( P(O) \) is just a normalizing constant.

- How to obtain \( P(O | L) \)? \( \Rightarrow \) 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 $P(L_n \mid O^n)$

• Assumption: $P(O_t \mid L_t)$ 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 $P(L_t \mid L_{t-1})$ can be easily determined from the physical properties of the moving object
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
Back to Hidden Markov Models

- For inference, easier to think of as a long chain of variables
- (For learning, the two-state model more fitting)
Joint probability of a HMM

- Joint probability factorizes like a BN
  - HMM is a Bayesian network!
  \[
P(X_0, X_1, E_1, X_2, E_2, ..., X_t, E_t) = P(X_0) \prod_{i=1}^{t} P(X_i | X_{i-1}) P(E_i | X_i)
\]

- 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: \(P(X_{1:t} | e_{1:t})\)
Calculating $P(X_t \mid e_{1:t})$ in HMM

- Let's shoot for a recursive formula:

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

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

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

- and

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

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

$$= \sum_{x_t} P(X_{t+1} \mid x_t) P(x_t \mid e_{1:t})$$
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: $P(X_{t+k} \mid e_{1:t})$, $k>0$

- $P(X_{t+1} \mid e_{1:t})$ part of the forward algorithm
- and from that on evidence does not count, and one can just calculate forward:

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

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

\[
P(X_{t+3} \mid e_{1:t}) = \sum_{x_{t+2}} P(X_{t+3} \mid x_{t+2}, e_{1:t}) P(x_{t+2} \mid e_{1:t})
\]

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

- Obvious move: divide $e_{1:t}$ to $e_{1:k}$ and $e_{k+1:t}$.

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

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

- 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*)
The Viterbi algorithm

- 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:

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

- More:
  - see e.g. Russel & Norvig, Chapter 15.2.
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 Θ.
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
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 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 $e$ is improbable, generated random vectors seldom conform with $e$, thus it takes a long time before we get a good estimate $P(X|e)$.
  - With long $E$, all $e$ are improbable.

- So called likelihood weighting can alleviate the problem a little bit, but not enough.
\[ P \left( X \middle| \text{mb} \left( X \right) \right) \]

\[ P(X|mb(X)) \]

\[ = P(X|mb(x), Rest) \]

\[ = \frac{P(X, mb(X), Rest)}{P(mb(X), Rest)} \]

\[ \propto P(\text{All}) \]

\[ = \prod_{X_i \in X} P(X_i|Pa(X_i)) \]

\[ = P(X|Pa(X)) \prod_{C \in \text{ch}(X)} P(C|Pa(C)) \prod_{R \in \text{Rest} \cup Pa(V)} P(R|Pa(R)) \]

\[ \propto P(X|Pa(X)) \prod_{C \in \text{ch}(X)} P(C|Pa(C)) \]
Gibbs sampling

- 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 True:
  
  for \( V \) in \( \mathbf{X}, \mathbf{Y} \):
  
    generate \( v \) from \( P(V | \text{MarkovBlanket}(V)) \)
    replace \( v \) in \( \mathbf{x}, \mathbf{y} \).
    \[
    N[\mathbf{x}] += 1
    \]
  
    print normalize\( (N[\mathbf{x}]) \)
Why does it work

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

\[
P^*(\mathbf{z})q(\mathbf{z} \rightarrow \mathbf{z}') = P(\mathbf{z}|\mathbf{e})P(v'|\mathbf{z}_{-V}, \mathbf{e})
\]
\[
= P(v, \mathbf{z}_{-V}|\mathbf{e})P(v'|\mathbf{z}_{-V}, \mathbf{e})
\]
\[
= P(v|\mathbf{z}_{-V}, \mathbf{e})P(\mathbf{z}_{-V}|\mathbf{e})P(v'|\mathbf{z}_{-V}, \mathbf{e})
\]
\[
= P(v|\mathbf{z}_{-V}, \mathbf{e})P(v', \mathbf{z}_{-V}|\mathbf{e}) = q(\mathbf{z}' \rightarrow \mathbf{z})P^*(\mathbf{z}')\), 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
So let us play....

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