Learning Bayesian Networks (part 1)

Mark Craven and David Page
Computer Sciences 760

Spring 2019

Goals for the lecture

you should understand the following concepts

- the Bayesian network representation
- inference by enumeration
- Markov chain Monte Carlo (MCMC) and Gibbs Sampling
- the parameter learning task for Bayes nets
- the structure learning task for Bayes nets
- maximum likelihood estimation
- Laplace estimates
- *m*-estimates
- missing data in machine learning
    - hidden variables
    - missing at random
    - missing systematically
- the EM approach to imputing missing values in Bayes net parameter learning

Bayesian network example

- Consider the following 5 binary random variables:

  B = a burglary occurs at your house

  E = an earthquake occurs at your house

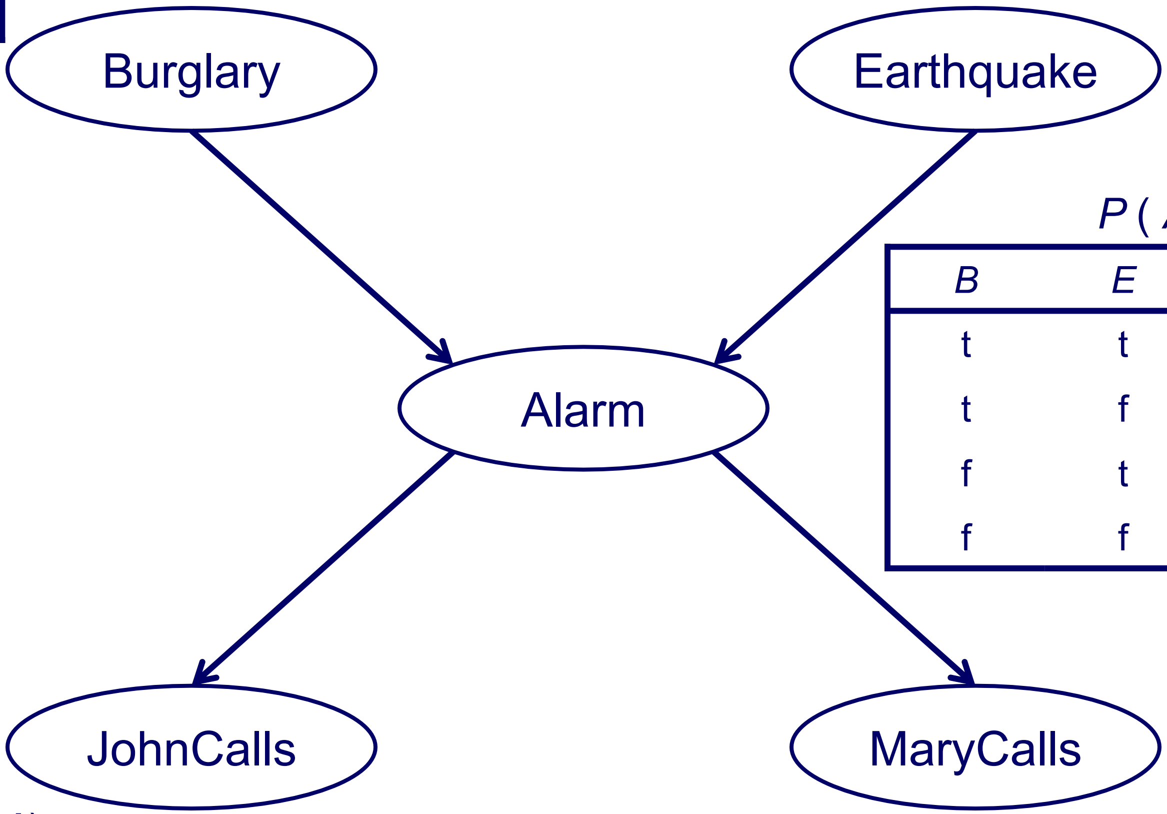
  A = the alarm goes off

  J = John calls to report the alarm

  M = Mary calls to report the alarm

- Suppose we want to answer queries like what is $P(B \mid M, J)$?

Bayesian network example



P (*B*)

t	f
0.001	0.999

P (*E*)

t	f
0.001	0.999

P (*A* \| *B*, *E*)

B	*E*	t	f
t	t	0.95	0.05
t	f	0.94	0.06
f	t	0.29	0.71
f	f	0.001	0.999

P (*J* \| *A*)

A	t	f
t	0.9	0.1
f	0.05	0.95

P (*M* \| *A*)

A	t	f
t	0.7	0.3
f	0.01	0.99

Bayesian networks

- a BN consists of a Directed Acyclic Graph (DAG) and a set of conditional probability distributions

- in the DAG
  - each node denotes random a variable
  - each edge from X to Y represents that X *directly influences* Y
  - formally: each variable X is independent of its non-descendants given its parents

- each node X has a *conditional probability distribution* (CPD) representing $P(X \mid Parents(X)\ )$

Bayesian networks

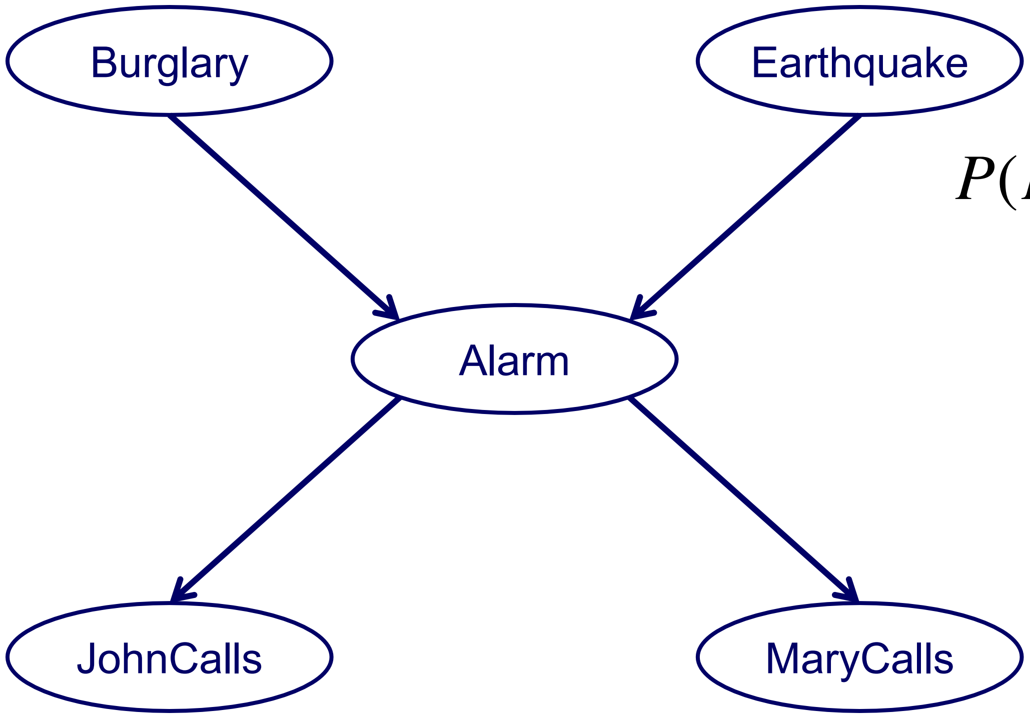
- using the chain rule, a joint probability distribution can be expressed as

$$P(X_1, \ldots, X_n) = \mathrm{P}(X_1) \prod_{i=2}^{n} P(X_i \mid X_1, \ldots, X_{i-1})$$

- a BN provides a compact representation of a joint probability distribution

$$P(X_1, \ldots, X_n) = \prod_{i=1}^{n} P(X_i \mid Parents(X_i))$$

Bayesian networks



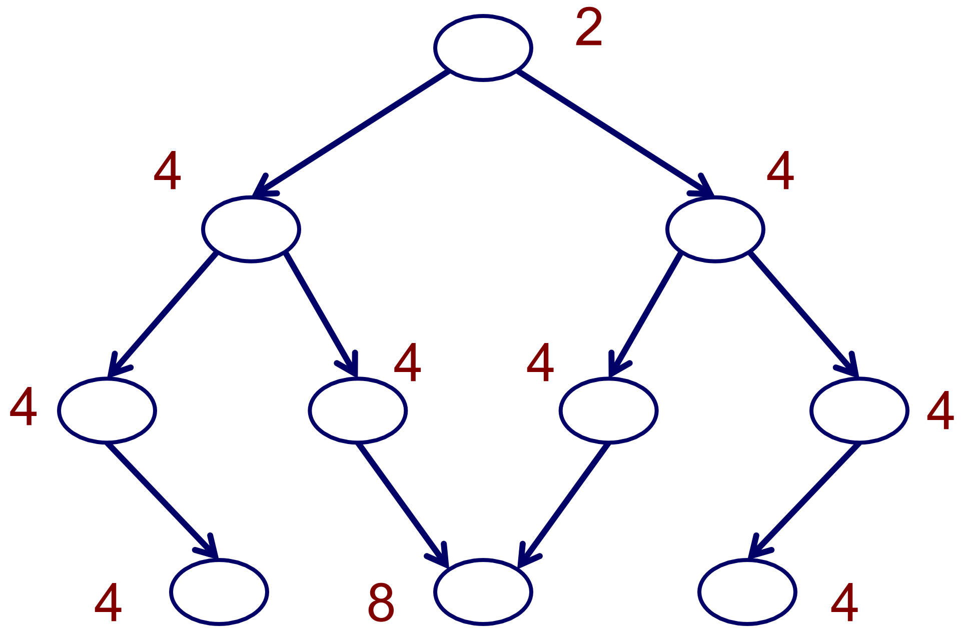
$$
\begin{aligned}
P(B,E,A,J,M) = \; & P(B)\times \\
& P(E)\times \\
& P(A \mid B,E)\times \\
& P(J \mid A)\times \\
& P(M \mid A)
\end{aligned}
$$

- a standard representation of the joint distribution for the Alarm example has $2^5 = 32$ parameters
- the BN representation of this distribution has 20 parameters

Bayesian networks

- consider a case with 10 binary random variables

- How many parameters does a BN with the following graph structure have?



= 42

- How many parameters does the standard table representation of the joint distribution have?

= 1024

Advantages of the Bayesian network representation

- Captures independence and conditional independence where they exist
- Encodes the relevant portion of the full joint among variables where dependencies exist
- Uses a graphical representation which lends insight into the complexity of inference

The inference task in Bayesian networks

Given: values for some variables in the network (*evidence*), and a set of *query* variables

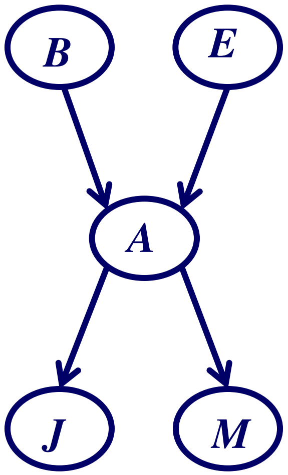
Do: compute the posterior distribution over the query variables

- variables that are neither evidence variables nor query variables are *hidden* variables
- the BN representation is flexible enough that any set can be the evidence variables and any set can be the query variables

Inference by enumeration

- let a denote A=true, and $\neg a$ denote A=false
- suppose we're given the query: $P(b \mid j, m)$

  "probability the house is being burglarized given that John and Mary both called"
- from the graph structure we can first compute:



$$P(b, j, m) = \sum_{e, \neg e} \sum_{a, \neg a} P(b)P(E)P(A \mid b, E)P(j \mid A)P(m \mid A)$$

sum over possible values for E and A variables (e, $\neg e$, a, $\neg a$)

Inference by enumeration

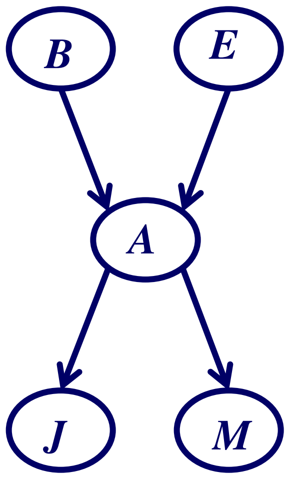
$$P(b,j,m) = \sum_{e,\neg e}\sum_{a,\neg a} P(b)P(E)P(A \mid b,E)P(j \mid A)P(m \mid A)$$

$$= P(b)\sum_{e,\neg e}\sum_{a,\neg a} P(E)P(A \mid b,E)P(j \mid A)P(m \mid A)$$

P(B)
0.001

P(E)
0.001

B	E	P(A)
t	t	0.95
t	f	0.94
f	t	0.29
f	f	0.001



A	P(J)
t	0.9
f	0.05

A	P(M)
t	0.7
f	0.01

$B \quad E \quad A \quad J \quad M$

$$\begin{aligned} &= 0.001 \times (0.001 \times 0.95 \times 0.9 \times 0.7 + && e, a \\ &\quad 0.001 \times 0.05 \times 0.05 \times 0.01 + && e, \neg a \\ &\quad 0.999 \times 0.94 \times 0.9 \times 0.7 + && \neg e, a \\ &\quad 0.999 \times 0.06 \times 0.05 \times 0.01) && \neg e, \neg a \end{aligned}$$

Inference by enumeration

- now do equivalent calculation for $P(\neg b, j, m)$
- and determine $P(b \mid j, m)$

$$P(b \mid j,m) = \frac{P(b,j,m)}{P(j,m)} = \frac{P(b,j,m)}{P(b,j,m) + P(\neg b,j,m)}$$

Really?

- Inference by Enumeration is more or less building the full joint table (at least, the part consistent with the evidence), summing out all but the query variable(s), and normalizing what remains to sum to 1.0

- Bayes net's factored representation didn't save us much

- In practice, this is unrealistic and we do inference differently – let's just look at one real approach…

Approximate (Monte Carlo) Inference in Bayes Nets

- Basic idea: Let’s repeatedly sample according to the distribution represented by the Bayes Net. If in 400/1000 draws, the variable *X* is *true*, then we estimate that the probability *X* is *true* is 0.4.
- To sample according to Bayes Net, just set the variables one at a time using a total ordering consistent with the partial...

Monte Carlo (continued)

- (Samping continued)… ordering represented by the underlying DAG of the Net. In this way, when we wish to draw the value for *X* we already have the values of its parents, so we can find the probabilities to use from the CPT for *X*.
- This approach is simple to implement using a pseudorandom number generator.

So it seems we’re done, right?

- Wrong: what if we take into account evidence (observed values for variables)?
- If the evidence happens to be in the “top” nodes of the network (nodes with no parents), we’re still fine. Otherwise...
- No efficient general method exists for sampling according to the new distribution based on the evidence. (There are inefficient ways, e.g., compute full joint.)

Rejection Sampling

- One natural option for sampling with evidence is to use our original sampling approach, and just throw out (*reject*) any setting that does not agree with the evidence. This is *rejection sampling*.
- Problem: if evidence involves many variables, most of our draws will be rejected (few will agree with the evidence).

Likelihood Weighting

- Another approach is to set the evidence variables, sample the others with the original Monte Carlo approach, and then correct for improbable combinations by *weighting* each setting by its probability.
- Disadvantage: with many evidence variables, probabilities become vanishingly small. We don’t sample the more probable events very thoroughly.

Markov Chain Monte Carlo

- Key idea: give up on *independence* in sampling.
- Generate next setting probabilistically based on current setting (Markov chain).
- Metropolis-Hastings Algorithm for the general case, Gibbs Sampler for Bayes Nets in particular. Key property: *detailed balance* yields *stationary distribution*.

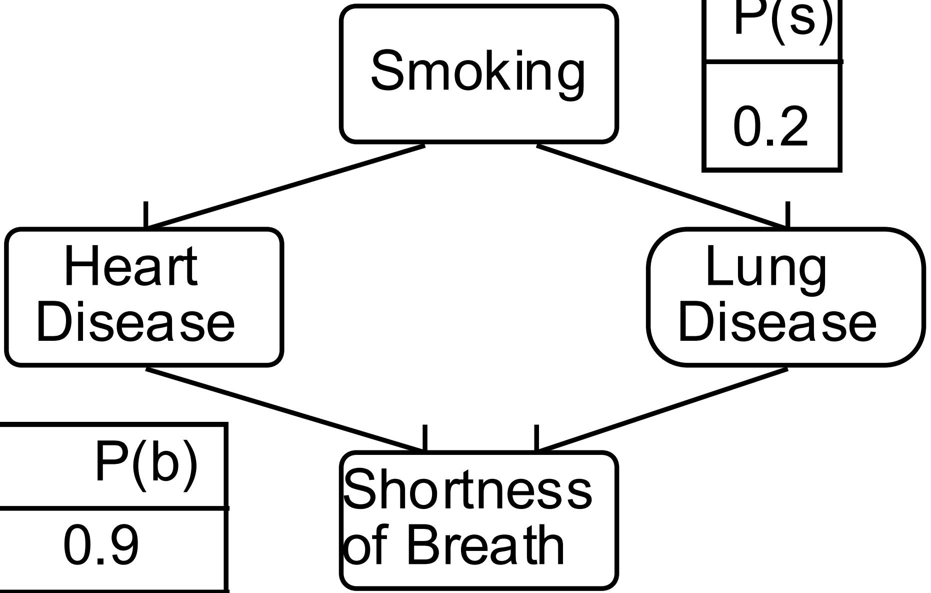
MCMC: Gibbs Sampling

```
$state$ ← random assignment to non-evidence variables
for $i$ ← 1 to $num\text{-}samples$ do
    for each non-evidence variable $x$
        sample $x$ according to $P(x|Parents(x))$
        $state$ ← $state$ with new value of $x$
$P(F)$ ← fraction of states in which $F$ is true
```

Gibbs Sampling by Example

S	P(h)
T	0.6
F	0.1

P(s)
0.2



S	P(l)
T	0.8
F	0.1

H	L	P(b)
T	T	0.9
T	F	0.8
F	T	0.7
F	F	0.1

Gibbs Sampling Example (Continued)

- Let our query be **P**(*HeartDisease* | *smoking, shortnessOfBreath*). That is, we know we've been smoking (*Smoking=True)* and we know we're experiencing shortness of breath (*ShortnessOfBreath=True*), and we wish the know the probability that we have heart disease.
- Might as well keep a tally for *LungDisease* while we're at it.

Other Decisions

- Let’s assume we use an off-the-shelf pseudorandom number generator for the range [0..1].
- We can loop through the non-evidence variables in a pre-established order or randomly, uniformly. Let’s go with the latter. Tally at each step. (If the former, we could tally at each step or each iteration.)

Other Decisions (Continued)

- One chain or many: let’s go with one.
- Length or burn-in: ordinarily 500-1000, but let’s go with 2 (don’t tally for original setting or setting after first step).
- Initial values: let’s say all *True*. Note that *Smoking* and *ShortnessOfBreath* must be initialized to *True*, since this is our evidence. The initial settings for non-evidence variables are arbitrary.

Other Decisions (Continued)

- Use of random numbers in selecting a variable to draw. We have only two non-evidence variables: *HeartDisease* and *LungDisease*. Let's adopt a policy that a random number greater than 0.5 leads us to draw a value for *LungDisease*, and a random number of 0.5 or less leads us to draw for *HeartDisease*.

Other Decisions (Continued)

- Use of random numbers in selecting values of variables. Since all our variables are Boolean, our distributions will be over the values <*True,*False> and will have the form <P(*True*),1-P(*True*)>. If our random number is less than or equal to P(*True*), then we will set the variable to *True*, and otherwise we will set it to *False*.

A Final Supposition for our Example

- Having made all our choices, the only other factor that will affect the activity of the Gibbs Sampling algorithm is the sequence of random numbers that we draw.
- Let’s suppose our sequence of random numbers begins 0.154, 0.383, 0.938, 0.813, 0.273, 0.739, 0.049, 0.233, 0.743, 0.932, 0.478, 0.832, …

Round 1

- Our first random number is 0.154, so we will draw a value for *HeartDisease*.
- To draw the value, we must first determine the distribution for *HeartDisease* given its Markov Blanket.
- First, we compute a value for *True*. We multiply P(*heartDisease*|*smoking*) by P(*shortnessOfBreath* | *heartDisease, lungDisease*). Notice we take *LungDisease*

Round 1 (Continued)

- (Continued)… to be *True* because that is its current setting. (We use the current settings of all variables in the Markov Blanket.) This product is (0.6)(0.9) = 0.54.
- Next we repeat the process for *HeartDisease=False*. We multiply the probability that *HeartDisease* is *False* given *smoking* by the probability of *shortnessOfBreath* given *HeartDisease* is

Round 1 (Continued)

- (Continued)… *False* and *LungDisease* is *True*. The resulting product is (0.4)(0.7) = 0.28.
- We now normalize <0.54,0.28> to get the probability distribution <0.66,0.34>. Hence we will set *HeartDisease* to *True* if and only if our random number is at most 0.66. It is 0.383, so *HeartDisease* remains *True*.

Round 2

- Our next random number is 0.938, so we next will draw a value for *LungDisease* given the current settings for the other variables.
- To obtain a value for *LungDisease=True*, we multiply P(*lungDisease* | *smoking*) by P(*shortnessOfBreath* | *heartDisease, lungDisease*). (Recall that *True* is our current setting for *HeartDisease* and *True*

Round 2 (Continued)

- (Continued)… is our candidate setting for *LungDisease*. This product is (0.8)(0.9) = 0.72.
- Similarly, for *LungDisease=False*, we multiply P(*LungDisease=False* | *smoking*) by P(*shortnessOfBreath* | *heartDisease, LungDisease=False*). This product is (0.2)(0.8) = 0.16.

Round 2 (Continued)

- Normalizing <0.72, 0.16> we get the distribution <0.82, 0.18>.
- Our next random number is 0.813, so we (barely) keep *LungDisease* set to *True.*
- This is the first round after our burn-in, so we record the frequencies. We now have counts of 0 for *HeartDisease* and *LungDisease* set to *False*, and counts of 1 for each of these set to *True*.

Round 3

- Our next random number is 0.273, so we draw a value for *HeartDisease* next.
- Because all the variables have the same value as the last time we drew for *HeartDisease*, the distribution is the same: <0.66, 0.34>. Our next random number is 0.739, so we set *HeartDisease* to *False.*

Round 3 (Continued)

- Updating our tallies, we have counts of: 1 for *HeartDisease=False,* 1 for *HeartDisease=True*, 0 for *LungDisease=False*, and 2 for *LungDisease=True.*

Round 4

- The next random number is 0.049. Therefore we draw a value for *HeartDisease* again. Because all other variables are unchanged, and we consider both values of *HeartDisease*, once again the distribution is <0.66, 0.34>. Our next random number is 0.233, so we reset *HeartDisease* to *True*.

Round 4 (Continued)

- Our new counts are as follows: 2 for *HeartDisease=True,* 1 for *HeartDisease=False,* 3 for *LungDisease=True*, and 0 for *LungDisease=False.*

Round 5

- Our next random number is 0.743, so we next draw a value for *LungDisease*.
- The values for all other variables are as they were the first time we drew a value for *LungDisease*, so the distribution remains <0.82,0.18>. Our next random number is 0.932, so we set *LungDisease* to *False*.

Round 5 (Continued)

- Our new tallies are as follows: 3 each for *HeartDisease=True* and *LungDisease=True*, and 1 each for *HeartDisease=False* and *LungDisease=False.*

Round 6

- The next random number is 0.478, so again we sample *HeartDisease.* But since the setting for *LungDisease* has changed, we must recompute the distribution over *HeartDisease.*
- To get a value for *HeartDisease=True*, we multiply P(*heartDisease* | *smoking*) by P(*shortnessOfBreath* | *HeartDisease=True,*

Round 6 (Continued)

- (Continued)… *LungDisease=False*). This results in the product (0.6)(0.8) = 0.48.
- For *HeartDisease=False*, we multiply P(*HeartDisease=False* | *smoking*) by P(*shortnessOfBreath* | *HeartDisease=False, LungDisease=False*). The result is (0.4)(0.1) = 0.04.
- Normalizing these values to obtain a

Round 6 (Continued)

- (Continued)… probability distribution, we get <0.92, 0.08>. Our next random number is 0.832 so we choose *HeartDisease=True.*
- Our tallies now stand at 1 for *HeartDisease=False,* 4 for *HeartDisease=True,* 2 for *LungDisease=False,* and 3 for *LungDisease=True.*

Final Results

- Of course, we have not run the Markov chain nearly long enough to expect an accurate estimate. Nevertheless, let’s ask what the answer is to our query at this point.
- We assign a probability of 4/5 or 0.8 to *heartDisease.*
- We also might ask about *lungDisease*, to which we assign 3/5 or 0.6.

Markov Chains: A 10-Slide Theoretical Detour

- A Markov chain is a process defined by:
  - A set of states
  - A set of associated transition probabilities
    - For every pair of states *s* and *s*′ (not necessarily distinct) we have an associated transition probability T(*s*➔*s*′) of moving from state *s* to state *s*′
    - For any time *t*, T(*s*➔*s*′) is the probability of the Markov process being in state *s*′ at time *t+1* given that it is in state *s* at time *t*

Some Properties of Markov Chains

- **Irreducible** chain: can get from any state to any other eventually (non-zero probability)
- **Periodic** state: state *i* is periodic with period *k* if all returns to *i* must occur in multiples of *k*
- **Ergodic** chain: irreducible and has an aperiodic state. Implies all states are aperiodic, so chain is aperiodic.
- Finite state space: can represent chain as matrix of transition probabilities… then *ergodic* = *regular*…
- **Regular** chain: some power of chain (transition matrix) has only positive elements
- **Reversible** chain: satisfies detailed balance (**later**)

Sufficient Condition for Regularity

- A Markov chain is regular if the following properties both hold:

  1. For any pair of states *s*, *s'* that each have nonzero probability there exists some path from *s* to *s'* with nonzero probability

  2. For all *s* with nonzero probability, the "self loop" probability T(*s*→*s*) is nonzero

- Gibbs sampling is regular if no zeroes in CPTs

Notation

- $\pi_t(\boldsymbol{y})$ = probability of being in state ***y*** at time *t*

- Transition function $T(\boldsymbol{y} \rightarrow \boldsymbol{y}')$ = probability of moving from state ***y*** to state ***y'***

How π Changes with Time in Markov Chain

- $\pi_{t+1}(\boldsymbol{y}') = \sum_{\mathbf{y}} \pi_t(\mathbf{y}) T(\mathbf{y} \to \mathbf{y}')$
- A distribution π_t is stationary if $\pi_t = \pi_{t+1}$, that is, for all $\boldsymbol{y}$, $\pi_t(\boldsymbol{y}) = \pi_{t+1}(\boldsymbol{y})$

Detailed Balance

- A Markov chain satisfies detailed balance if there exists a unique distribution π such that for all states $\boldsymbol{y}$, $\boldsymbol{y'}$,

$$\pi(\boldsymbol{y})T(\boldsymbol{y} \rightarrow \boldsymbol{y'}) = \pi(\boldsymbol{y'})T(\boldsymbol{y'} \rightarrow \boldsymbol{y})$$

- If a regular Markov chain satisfies detailed balance with distribution π, then there exists *t* such that for any initial distribution π_0, $\pi_t = \pi$
- Detailed balance with regularity implies convergence to unique stationary distribution

Gibbs Sampler satisfies Detailed Balance

A Gibbs sampler Markov chain defined by a Bayesian network with all CPT entries nonzero, representing probability distribution P, satisfies detailed balance with probability distribution $\pi(\boldsymbol{y})=P(\boldsymbol{y})$ for all states $\boldsymbol{y}$

Is special case of Metropolis-Hastings Algorithm, next

Using Other Samplers

- The Gibbs sampler only changes one random variable at a time
  - Slow convergence
  - High-probability states may not be reached because reaching them requires going through low-probability states
  - If zeroes in some CPTs, may fail to achieve detailed balance

Metropolis Sampler

- Propose a transition with probability $T^Q(\boldsymbol{y} \rightarrow \boldsymbol{y'})$
- Accept with probability $A=\min(1, P(\boldsymbol{y'})/P(\boldsymbol{y}))$
- If for all $\boldsymbol{y}$, $\boldsymbol{y'}$ $T^Q(\boldsymbol{y} \rightarrow \boldsymbol{y'})=T^Q(\boldsymbol{y'} \rightarrow \boldsymbol{y})$ then the resulting Markov chain satisfies detailed balance

Metropolis-Hastings Sampler

- Propose a transition with probability $T^Q(\boldsymbol{y} \rightarrow \boldsymbol{y'})$
- Accept with probability
  $A=\min(1, P(\boldsymbol{y'})T^Q(\boldsymbol{y'} \rightarrow \boldsymbol{y})/P(\boldsymbol{y})T^Q(\boldsymbol{y} \rightarrow \boldsymbol{y'}))$
- Detailed balance satisfied
- Acceptance probability often easy to compute even though sampling according to P difficult

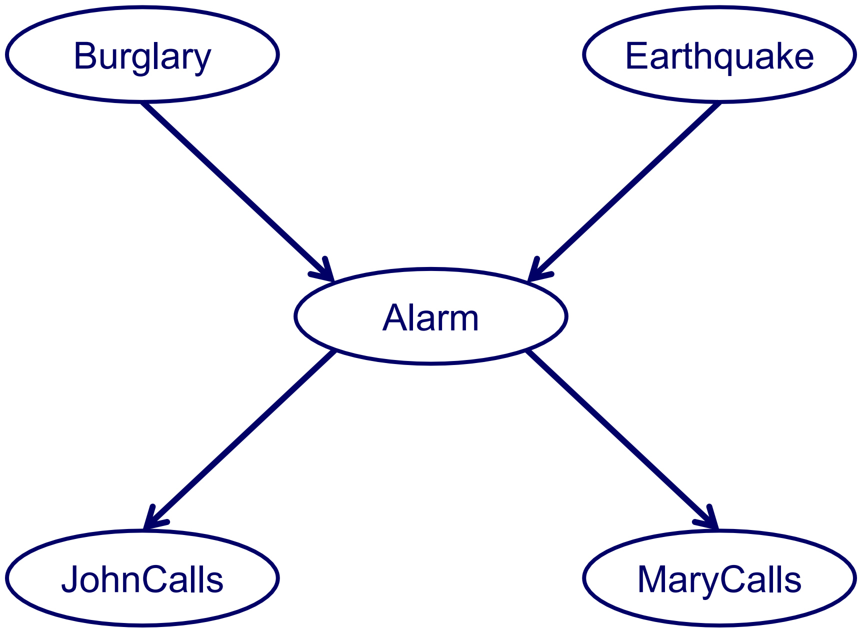
Comments on BN inference

- *inference by enumeration* is an *exact* method (i.e. it computes the exact answer to a given query)
- it requires summing over a joint distribution whose size is exponential in the number of variables
- in many cases we can do exact inference efficiently in large networks
  - key insight: save computation by pushing sums inward
- in general, the Bayes net inference problem is NP-hard
- other exact methods: variable elimination, junction trees
- there are also methods for approximate inference – these get an answer which is “close”: loopy belief propagation, MCMC (next)
- in general, the approximate inference problem is NP-hard also, but approximate methods work well for many real-world problems

The parameter learning task

- Given: a set of training instances, the graph structure of a BN

B	E	A	J	M
f	f	f	t	f
f	t	f	f	f
f	f	t	f	t
		…		



- Do: infer the parameters of the CPDs

The structure learning task

- Given: a set of training instances

B	E	A	J	M
f	f	f	t	f
f	t	f	f	f
f	f	t	f	t
		…		

- Do: infer the graph structure (and perhaps the parameters of the CPDs too)

Parameter learning and maximum likelihood estimation

- *maximum likelihood estimation* (MLE)
  - given a model structure (e.g. a Bayes net graph) G and a set of data D
  - set the model parameters θ to maximize $P(D \mid G, \theta)$

- i.e. make the data D look <u>as likely as possible</u> under the model $P(D \mid G, \theta)$

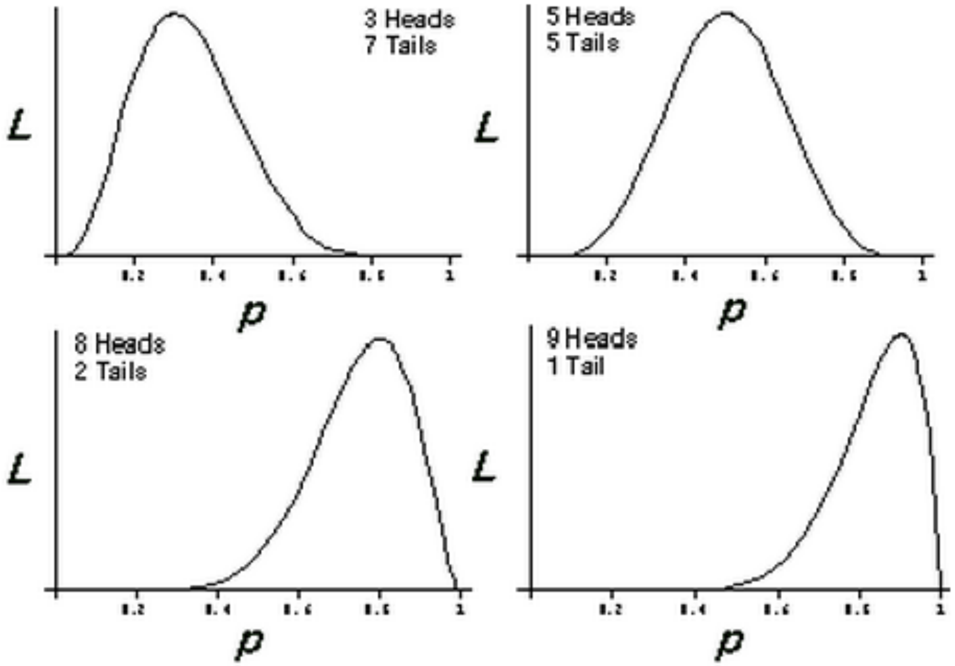
Maximum likelihood estimation

consider trying to estimate the parameter θ (probability of heads) of a biased coin from a sequence of flips

$$\boldsymbol{x} = \{1,1,1,0,1,0,0,1,0,1\}$$

the likelihood function for θ is given by:

$$L(\theta : x_1,\ldots,x_n) = \theta^{x_1}(1-\theta)^{1-x_1}\cdots\theta^{x_n}(1-\theta)^{1-x_n}$$
$$= \theta^{\sum x_i}(1-\theta)^{n-\sum x_i}$$



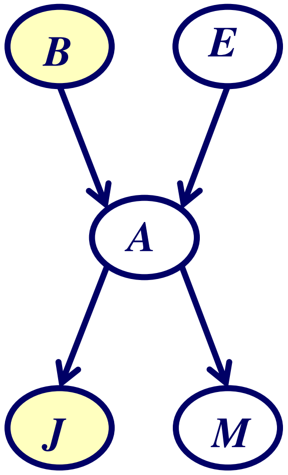
for h heads in n flips
the MLE is h/n

MLE in a Bayes net

$$
\begin{aligned}
L(\theta : D, G) = P(D \mid G, \theta) &= \prod_{d \in D} P(x_1^{(d)}, x_2^{(d)}, \ldots, x_n^{(d)}) \\
&= \prod_{d \in D} \prod_{i} P(x_i^{(d)} \mid Parents(x_i^{(d)})) \\
&= \prod_{i} \underbrace{\left(\prod_{d \in D} P(x_i^{(d)} \mid Parents(x_i^{(d)})) \right)}_{\text{independent parameter learning problem for each CPD}}
\end{aligned}
$$

Maximum likelihood estimation

now consider estimating the CPD parameters for *B* and *J* in the alarm network given the following data set



B	*E*	*A*	*J*	*M*
f	f	f	t	f
f	t	f	f	f
f	f	f	t	t
t	f	f	f	t
f	f	t	t	f
f	f	t	f	t
f	f	t	t	t
f	f	t	t	t

$$P(b) = \frac{1}{8} = 0.125$$

$$P(\neg b) = \frac{7}{8} = 0.875$$

$$P(j \mid a) = \frac{3}{4} = 0.75$$

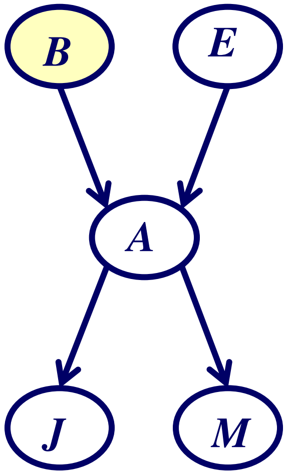
$$P(\neg j \mid a) = \frac{1}{4} = 0.25$$

$$P(j \mid \neg a) = \frac{2}{4} = 0.5$$

$$P(\neg j \mid \neg a) = \frac{2}{4} = 0.5$$

Maximum likelihood estimation

suppose instead, our data set was this…



B	E	A	J	M
f	f	f	t	f
f	t	f	f	f
f	f	f	t	t
f	f	f	f	t
f	f	t	t	f
f	f	t	f	t
f	f	t	t	t
f	f	t	t	t

$$\boxed{P(b) = \frac{0}{8} = 0}$$

$$P(\neg b) = \frac{8}{8} = 1$$

do we really want to set this to 0?

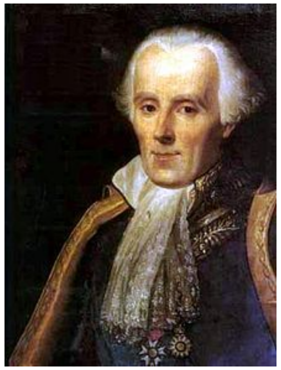
Maximum a posteriori (MAP) estimation

- instead of estimating parameters strictly from the data, we could start with some prior belief for each

- for example, we could use *Laplace estimates*

$$P(X=x)=\frac{n_x+1}{\sum_{v\in \text{Values}(X)}(n_v+1)}$$

pseudocounts



- where n_v represents the number of occurrences of value v

Maximum a posteriori estimation

a more general form: *m-estimates*

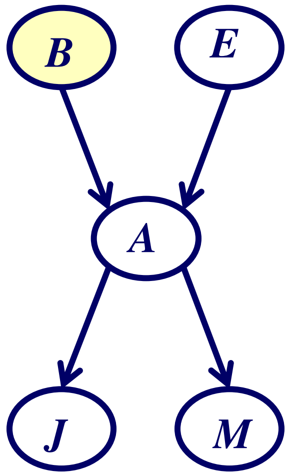
$$P(X = x) = \frac{n_x + p_x m}{\left(\sum_{v \in \mathrm{Values}(X)} n_v\right) + m}$$

prior probability of value x

number of "virtual" instances

M-estimates example

now let's estimate parameters for B using $m=4$ and $p_b=0.25$



B	*E*	*A*	*J*	*M*
f	f	f	t	f
f	t	f	f	f
f	f	f	t	t
f	f	f	f	t
f	f	t	t	f
f	f	t	f	t
f	f	t	t	t
f	f	t	t	t

$$P(b) = \frac{0 + 0.25 \times 4}{8+4} = \frac{1}{12} = 0.08 \qquad P(\neg b) = \frac{8 + 0.75 \times 4}{8+4} = \frac{11}{12} = 0.92$$

Missing data

- Commonly in machine learning tasks, some feature values are missing

- some variables may not be observable (i.e. *hidden*) even for training instances

- values for some variables may be *missing at random*: what caused the data to be missing does not depend on the missing data itself
  - e.g. someone accidentally skips a question on an questionnaire
  - e.g. a sensor fails to record a value due to a power blip

- values for some variables may be *missing systematically*: the probability of value being missing depends on the value
  - e.g. a medical test result is missing because a doctor was fairly sure of a diagnosis given earlier test results
  - e.g. the graded exams that go missing on the way home from school are those with poor scores

Missing data

- hidden variables; values *missing at random*
  - these are the cases we'll focus on
  - one solution: try impute the values

- values *missing systematically*
  - may be sensible to represent "*missing*" as an explicit feature value

Imputing missing data with EM

Given:

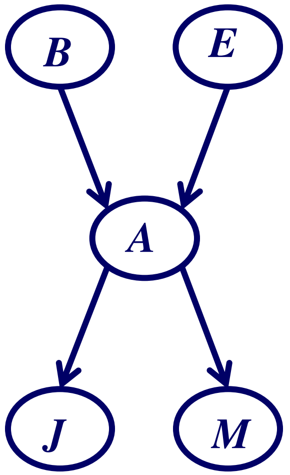
- data set with some missing values
- model structure, initial model parameters

Repeat until convergence

- *Expectation* (E) step: using current model, compute expectation over missing values
- *Maximization* (M) step: update model parameters with those that maximize probability of the data (MLE or MAP)

example: EM for parameter learning

suppose we're given the following <u>initial</u> BN and training set



P(B)
0.1

P(E)
0.2

B	E	P(A)
t	t	0.9
t	f	0.6
f	t	0.3
f	f	0.2

A	P(J)
t	0.9
f	0.2

A	P(M)
t	0.8
f	0.1

B	E	A	J	M
f	f	?	f	f
f	f	?	t	f
t	f	?	t	t
f	f	?	f	t
f	t	?	t	f
f	f	?	f	t
t	t	?	t	t
f	f	?	f	f
f	f	?	t	f
f	f	?	f	t

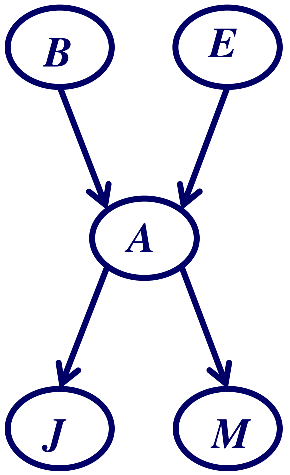
example: E-step

$$P(a \mid \neg b, \neg e, \neg j, \neg m)$$

$$P(\neg a \mid \neg b, \neg e, \neg j, \neg m)$$

P(B)
0.1

P(E)
0.2



B	*E*	*P(A)*
t	t	0.9
t	f	0.6
f	t	0.3
f	f	0.2

A	*P(J)*
t	0.9
f	0.2

A	*P(M)*
t	0.8
f	0.1

B	*E*	*A*	*J*	*M*
f	f	t: 0.0069 f: 0.9931	f	f
f	f	t:0.2 f:0.8	t	f
t	f	t:0.98 f: 0.02	t	t
f	f	t: 0.2 f: 0.8	f	t
f	t	t: 0.3 f: 0.7	t	f
f	f	t:0.2 f: 0.8	f	t
t	t	t: 0.997 f: 0.003	t	t
f	f	t: 0.0069 f: 0.9931	f	f
f	f	t:0.2 f: 0.8	t	f
f	f	t: 0.2 f: 0.8	f	t

example: E-step

$$P(a \mid \neg b, \neg e, \neg j, \neg m) = \frac{P(\neg b, \neg e, a, \neg j, \neg m)}{P(\neg b, \neg e, a, \neg j, \neg m) + P(\neg b, \neg e, \neg a, \neg j, \neg m)}$$

$$= \frac{0.9 \times 0.8 \times 0.2 \times 0.1 \times 0.2}{0.9 \times 0.8 \times 0.2 \times 0.1 \times 0.2 \; + \; 0.9 \times 0.8 \times 0.8 \times 0.8 \times 0.9}$$

$$= \frac{0.00288}{.4176} = 0.0069$$

$$P(a \mid \neg b, \neg e, j, \neg m) = \frac{P(\neg b, \neg e, a, j, \neg m)}{P(\neg b, \neg e, a, j, \neg m) + P(\neg b, \neg e, \neg a, j, \neg m)}$$

$$= \frac{0.9 \times 0.8 \times 0.2 \times 0.9 \times 0.2}{0.9 \times 0.8 \times 0.2 \times 0.9 \times 0.2 \; + \; 0.9 \times 0.8 \times 0.8 \times 0.2 \times 0.9}$$

$$= \frac{0.02592}{.1296} = 0.2$$

$$P(a \mid b, \neg e, j, m) = \frac{P(b, \neg e, a, j, m)}{P(b, \neg e, a, j, m) + P(b, \neg e, \neg a, j, m)}$$

$$= \frac{0.1 \times 0.8 \times 0.6 \times 0.9 \times 0.8}{0.1 \times 0.8 \times 0.6 \times 0.9 \times 0.8 \; + \; 0.1 \times 0.8 \times 0.4 \times 0.2 \times 0.1}$$

$$= \frac{0.03456}{.0352} = 0.98$$

$$\vdots$$

example: M-step

re-estimate probabilities using expected counts

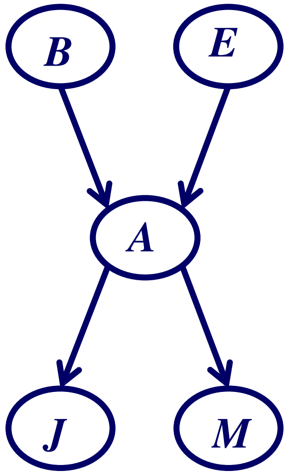
$$P(a \mid b,e) = \frac{E\#(a \wedge b \wedge e)}{E\#(b \wedge e)}$$

$$P(a \mid b,e) = \frac{0.997}{1}$$

$$P(a \mid b,\neg e) = \frac{0.98}{1}$$

$$P(a \mid \neg b,e) = \frac{0.3}{1}$$

$$P(a \mid \neg b,\neg e) = \frac{0.0069+0.2+0.2+0.2+0.0069+0.2+0.2}{7}$$



B	*E*	*P(A)*
t	t	0.997
t	f	0.98
f	t	0.3
f	f	0.145

re-estimate probabilities for $P(J \mid A)$ and $P(M \mid A)$ in same way

B	*E*	*A*	*J*	*M*
f	f	t: 0.0069 f: 0.9931	f	f
f	f	t:0.2 f:0.8	t	f
t	f	t:0.98 f: 0.02	t	t
f	f	t: 0.2 f: 0.8	f	t
f	t	t: 0.3 f: 0.7	t	f
f	f	t:0.2 f: 0.8	f	t
t	t	t: 0.997 f: 0.003	t	t
f	f	t: 0.0069 f: 0.9931	f	f
f	f	t:0.2 f: 0.8	t	f
f	f	t: 0.2 f: 0.8	f	t

example: M-step

re-estimate probabilities using expected counts

$$P(j \mid a) = \frac{E\#(a \wedge j)}{E\#(a)}$$

$$P(j \mid a) = \frac{0.2 + 0.98 + 0.3 + 0.997 + 0.2}{0.0069 + 0.2 + 0.98 + 0.2 + 0.3 + 0.2 + 0.997 + 0.0069 + 0.2 + 0.2}$$

$$P(j \mid \neg a) = \frac{0.8 + 0.02 + 0.7 + 0.003 + 0.8}{0.9931 + 0.8 + 0.02 + 0.8 + 0.7 + 0.8 + 0.003 + 0.9931 + 0.8 + 0.8}$$

B	E	A	J	M
f	f	t: 0.0069 f: 0.9931	f	f
f	f	t:0.2 f:0.8	t	f
t	f	t:0.98 f: 0.02	t	t
f	f	t: 0.2 f: 0.8	f	t
f	t	t: 0.3 f: 0.7	t	f
f	f	t:0.2 f: 0.8	f	t
t	t	t: 0.997 f: 0.003	t	t
f	f	t: 0.0069 f: 0.9931	f	f
f	f	t:0.2 f: 0.8	t	f
f	f	t: 0.2 f: 0.8	f	t

Convergence of EM

- E and M steps are iterated until probabilities converge
- will converge to a maximum in the data likelihood (MLE or MAP)
- the maximum may be a local optimum, however
- the optimum found depends on starting conditions (initial estimated probability parameters)