Outline

♦ Exact inference (briefly)
♦ Approximate inference (rejection sampling, MCMC)
♦ Parameter learning
Inference by enumeration

Slightly intelligent way to sum out variables from the joint without actually constructing its explicit representation

Simple query on the burglary network:
\[
P(B|j, m) = \frac{P(B, j, m)}{P(j, m)} = \alpha P(B, j, m) = \alpha \sum_e \sum_a P(B, e, a, j, m)
\]

Rewrite full joint entries using product of CPT entries:
\[
P(B|j, m) = \alpha \sum_e \sum_a P(B)P(e)P(a|B, e)P(j|a)P(m|a)
\]

Recursive depth-first enumeration: \(O(D)\) space, \(O(K^D)\) time
Evaluation tree

Enumeration is inefficient: repeated computation
e.g., computes $P(j|a)P(m|a)$ for each value of $e$
Efficient exact inference

Junction tree and variable elimination algorithms avoid repeated computation (generalized from of dynamic programming)

◊ **Singly connected** networks (or polytrees):
  - any two nodes are connected by at most one (undirected) path
  - time and space cost of exact inference are $O(K^L D)$

◊ **Multiply connected** networks:
  - can reduce 3SAT to exact inference $\Rightarrow$ NP-hard
  - equivalent to counting 3SAT models $\Rightarrow$ #P-complete

A B C D
1 2 3
AND
0.5

1. A v B v C
2. C v D v ¬A
3. B v C v ¬D
Inference by stochastic simulation

Idea: replace sum over hidden-variable assignments with a random sample.

1) Draw \( N \) samples from a sampling distribution \( S \)
2) Compute an approximate posterior probability \( \hat{P} \)
3) Show this converges to the true probability \( P \)

Outline:
- Sampling from an empty network
- Rejection sampling: reject samples disagreeing with evidence
- Markov chain Monte Carlo (MCMC): sample from a stochastic process whose stationary distribution is the true posterior
Sampling from an empty network

**function** `Prior-Sample(bn)` **returns** an event sampled from prior specified by `bn` inputs: `bn`, a Bayesian network specifying joint distribution `P(X_1, \ldots, X_D)`

- `x ←` an event with `D` elements
- **for** `j = 1, \ldots, D` **do**
  - `x[j] ←` a random sample from `P(X_j |` values of `parents(X_j)` in `x`)
- **return** `x`
Cloudy → Sprinkler → Wet Grass
Cloudy → Rain

| C | P(S|C) |
|---|-------|
| T | .10   |
| F | .50   |

| C | P(R|C) |
|---|-------|
| T | .80   |
| F | .20   |

| S | R | P(W|S,R) |
|---|---|---------|
| T | T | .99     |
| T | F | .90     |
| F | T | .90     |
| F | F | .01     |
Example

| C | P(S|C) |
|---|---|
| T | .10 |
| F | .50 |

<table>
<thead>
<tr>
<th>P(C)</th>
</tr>
</thead>
<tbody>
<tr>
<td>.50</td>
</tr>
</tbody>
</table>

| C | P(R|C) |
|---|---|
| T | .80 |
| F | .20 |

| S | R | P(W|S,R) |
|---|---|---|
| T | T | .99 |
| T | F | .90 |
| F | T | .90 |
| F | F | .01 |
Example

Cloudy

| C | P(S|C) |
|---|-------|
| T | .10   |
| F | .50   |

Sprinkler

| C | P(R|C) |
|---|-------|
| T | .80   |
| F | .20   |

Wet Grass

| S | R   | P(W|S,R) |
|---|-----|---------|
| T | T   | .99     |
| T | F   | .90     |
| F | T   | .90     |
| F | F   | .01     |

Rain

CS194-10 Fall 2011 Lecture 22
Example

| C | P(S|C) | P(R|C) | P(W|S,R) |
|---|-------|--------|----------|
| T | .10   | .80    | .99      |
| F | .50   | .20    | .90      |

Cloudy

Sprinkler

Wet Grass

Rain

P(C) = .50

P(S|C) = .10, .50

P(R|C) = .80, .20

P(W|S,R) = .99, .90, .90, .01
Example

\[
\begin{array}{c|c}
C & P(S|C) \\
T & .10 \\
F & .50 \\
\end{array}
\]

\[
\begin{array}{c|c|c|c|c}
S & R & P(W|S,R) \\
T & T & .99 \\
T & F & .90 \\
F & T & .90 \\
F & F & .01 \\
\end{array}
\]

\[
\begin{array}{c|c}
C & P(R|C) \\
T & .80 \\
F & .20 \\
\end{array}
\]
Example

| C | P(S|C) |
|---|-------|
| T | .10   |
| F | .50   |

| C | P(R|C) |
|---|-------|
| T | .80   |
| F | .20   |

| S | R | P(W|S,R) |
|---|---|---------|
| T | T | .99     |
| T | F | .90     |
| F | T | .90     |
| F | F | .01     |
Probability that $\text{PRIORSample}$ generates a particular event

$$S_{PS}(x_1 \ldots x_D) = \prod_{j=1}^{D} P(x_j | \text{parents}(X_j)) = P(x_1 \ldots x_D)$$

i.e., the true prior probability

E.g., $S_{PS}(t, f, t, t) = 0.5 \times 0.9 \times 0.8 \times 0.9 = 0.324 = P(t, f, t, t)$

Let $N_{PS}(x_1 \ldots x_D)$ be the number of samples generated for event $x_1, \ldots, x_D$

Then we have

$$\lim_{N \to \infty} \hat{P}(x_1, \ldots, x_D) = \lim_{N \to \infty} \frac{N_{PS}(x_1, \ldots, x_D)}{N} = S_{PS}(x_1, \ldots, x_D) = P(x_1 \ldots x_D)$$

That is, estimates derived from $\text{PRIORSample}$ are consistent

Shorthand: $\hat{P}(x_1, \ldots, x_D) \approx P(x_1 \ldots x_D)$
Rejection sampling

\( \hat{P}(X|e) \) estimated from samples agreeing with \( e \)

```plaintext
function REJECTION-SAMPLING(\( X, e, bn, N \)) returns an estimate of \( P(X|e) \)
local variables: \( N \), a vector of counts for each value of \( X \), initially zero

for \( i = 1 \) to \( N \) do
    \( x \leftarrow \text{PRIOR-SAMPLE}(bn) \)
    if \( x \) is consistent with \( e \) then
        \( N[x] \leftarrow N[x] + 1 \) where \( x \) is the value of \( X \) in \( x \)
return \( \text{NORMALIZE}(N) \)
```

E.g., estimate \( P(Rain|Sprinkler = true) \) using 100 samples
27 samples have \( Sprinkler = true \)
    Of these, 8 have \( Rain = true \) and 19 have \( Rain = false \).

\( \hat{P}(Rain|Sprinkler = true) = \text{NORMALIZE}(<8,19>) = <0.296, 0.704> \)

Similar to a basic real-world empirical estimation procedure
Analysis of rejection sampling

\[ \hat{P}(X|e) = \alpha N_{PS}(X, e) \quad \text{(algorithm defn.)} \]
\[ = N_{PS}(X, e)/N_{PS}(e) \quad \text{(normalized by } N_{PS}(e)) \]
\[ \approx P(X, e)/P(e) \quad \text{(property of PRIORSAMPLE)} \]
\[ = P(X|e) \quad \text{(defn. of conditional probability)} \]

Hence rejection sampling returns consistent posterior estimates

Problem: hopelessly expensive if \( P(e) \) is small

\( P(e) \) drops off exponentially with number of evidence variables!
Approximate inference using MCMC

General idea of Markov chain Monte Carlo

◊ Sample space $\Omega$, probability $\pi(\omega)$ (e.g., posterior given $e$)
◊ Would like to sample directly from $\pi(\omega)$, but it’s hard
◊ Instead, wander around $\Omega$ randomly, collecting samples
◊ Random wandering is controlled by transition kernel $\phi(\omega \rightarrow \omega')$ specifying the probability of moving to $\omega'$ from $\omega$
  (so the random state sequence $\omega_0, \omega_1, \ldots, \omega_t$ is a Markov chain)
◊ If $\phi$ is defined appropriately, the stationary distribution is $\pi(\omega)$
  so that, after a while (mixing time) the collected samples are drawn from $\pi$
Gibbs sampling in Bayes nets

Markov chain state $\omega_t = \text{current assignment } x_t$ to all variables

Transition kernel: pick a variable $X_j$, sample it conditioned on all others

Markov blanket property: $P(X_j | \text{all other variables}) = P(X_j | \text{mb}(X_j))$
so generate next state by sampling a variable given its Markov blanket

function GIBBS-ASK($X, e, bn, N$) returns an estimate of $P(X|e)$

local variables: $N$, a vector of counts for each value of $X$, initially zero
$Z$, the nonevidence variables in $bn$
$z$, the current state of variables $Z$, initially random

for $i = 1$ to $N$ do
  choose $Z_j$ in $Z$ uniformly at random
  set the value of $Z_j$ in $z$ by sampling from $P(Z_j|\text{mb}(Z_j))$
  $N[x] \leftarrow N[x] + 1$ where $x$ is the value of $X$ in $z$
return Normalize($N$)
With $Sprinkler = true, WetGrass = true$, there are four states:
MCMC example contd.

Estimate $P(Rain|Sprinkler = true, WetGrass = true)$

Sample $Cloudy$ or $Rain$ given its Markov blanket, repeat. Count number of times $Rain$ is true and false in the samples.

E.g., visit 100 states

31 have $Rain = true$, 69 have $Rain = false$

$\hat{P}(Rain|Sprinkler = true, WetGrass = true) = \text{Normalize}(\langle 31, 69 \rangle) = \langle 0.31, 0.69 \rangle$
Markov blanket sampling

Markov blanket of *Cloudy* is
*Sprinkler* and *Rain*

Markov blanket of *Rain* is
*Cloudy*, *Sprinkler*, and *WetGrass*

Probability given the Markov blanket is calculated as follows:

$$P(x'_j | mb(X_j)) = \alpha P(x'_j | parents(X_j)) \prod_{Z_\ell \in Children(X_j)} P(z_\ell | parents(Z_\ell))$$

E.g., \( \phi(\neg cloudy, rain \rightarrow cloudy, rain) \)

\[= 0.5 \times \alpha P(cloudy) P( | cloudy) P( | cloudy) = 0.5 \times \alpha \times 0.5 \times 0.1 \times 0.8 \]

\[= 0.5 \times \frac{0.040}{0.040+0.050} = 0.2222 \]

(Easy for discrete variables; continuous case requires mathematical analysis for each combination of distribution types.)

Easily implemented in message-passing parallel systems, brains

Can converge slowly, especially for near-deterministic models
Theory for Gibbs sampling

Theorem: stationary distribution for Gibbs transition kernel is $P(z \mid e)$; i.e., long-run fraction of time spent in each state is exactly proportional to its posterior probability

Proof sketch:
- The Gibbs transition kernel satisfies detailed balance for $P(z \mid e)$
  i.e., for all $z, z'$ the “flow” from $z$ to $z'$ is the same as from $z'$ to $z$
- $\pi$ is the unique stationary distribution for any ergodic transition kernel satisfying detailed balance for $\pi$
Let $\pi_t(z)$ be the probability the chain is in state $z$ at time $t$.

Detailed balance condition: “outflow” = “inflow” for each pair of states:

$$\pi_t(z) \phi(z \rightarrow z') = \pi_t(z') \phi(z' \rightarrow z)$$

for all $z, z'$.

Detailed balance $\Rightarrow$ stationarity:

$$\pi_{t+1}(z) = \sum_{z'} \pi_t(z') \phi(z' \rightarrow z) = \sum_{z'} \pi_t(z) \phi(z \rightarrow z')$$

$$= \pi_t(z) \sum_{z'} \phi(z \rightarrow z')$$

$$= \pi_t(z')$$

MCMC algorithms typically constructed by designing a transition kernel $\phi$ that is in detailed balance with desired $\pi$. 

---

CS194-10 Fall 2011 Lecture 22  24
Gibbs sampling transition kernel

Probability of choosing variable $Z_j$ to sample is $1/(D - |E|)$

Let $\bar{Z}_j$ be all other nonevidence variables, i.e., $Z - \{Z_j\}$

Current values are $z_j$ and $\bar{z}_j$; $e$ is fixed; transition probability is given by

$$\phi(z \to z') = \phi(z_j, \bar{z}_j \to z'_j, \bar{z}_j) = P(z'_j | \bar{z}_j, e)/(D - |E|)$$

This gives detailed balance with $P(z \mid e)$:

$$\pi(z)\phi(z \to z') = \frac{1}{D - |E|} P(z \mid e) P(z'_j | \bar{z}_j, e) = \frac{1}{D - |E|} P(z_j, \bar{z}_j \mid e) P(z'_j | \bar{z}_j, e)$$

$$= \frac{1}{D - |E|} P(z_j | \bar{z}_j, e) P(\bar{z}_j \mid e) P(z'_j | \bar{z}_j, e) \quad \text{(chain rule)}$$

$$= \frac{1}{D - |E|} P(z_j | \bar{z}_j, e) P(z'_j, \bar{z}_j \mid e) \quad \text{(chain rule backwards)}$$

$$= \phi(z' \to z) \pi(z') = \pi(z')\phi(z' \to z)$$
Summary (inference)

Exact inference:
- polytime on polytrees, NP-hard on general graphs
- space = time, very sensitive to topology

Approximate inference by MCMC:
- Generally insensitive to topology
- Convergence can be very slow with probabilities close to 1 or 0
- Can handle arbitrary combinations of discrete and continuous variables
Parameter learning: Complete data

\[ \theta_{jk\ell} = P(X_j = k \mid Parents(X_j) = \ell) \]

Let \( x_j^{(i)} = \text{value of } X_j \text{ in example } i \); assume Boolean for simplicity

Log likelihood

\[
L(\theta) = \sum_{i=1}^{N} \sum_{j=1}^{D} \log P(x_j^{(i)} \mid parents(X_j)^{(i)})
\]

\[
= \sum_{i=1}^{N} \sum_{j=1}^{D} \log \theta_{j1\ell(i)} x_j^{(i)} (1 - \theta_{j1\ell(i)})^{1-x_j^{(i)}}
\]

\[
\frac{\partial L}{\partial \theta_{j1\ell}} = \frac{N_{j1\ell}}{\theta_{j1\ell}} - \frac{N_{j0\ell}}{1 - \theta_{j1\ell}} = 0 \quad \text{gives}
\]

\[
\theta_{j1\ell} = \frac{N_{j1\ell}}{N_{j0\ell} + N_{j1\ell}} = \frac{N_{j1\ell}}{N_{j\ell}}.
\]

I.e., learning is completely decomposed; MLE = observed condition frequency
Example

Red/green wrapper depends probabilistically on flavor:

Likelihood for, e.g., cherry candy in green wrapper:

\[
P(F = \text{cherry}, W = \text{green}| h_{\theta, \theta_1, \theta_2})
= P(F = \text{cherry}| h_{\theta, \theta_1, \theta_2})P(W = \text{green}| F = \text{cherry}, h_{\theta, \theta_1, \theta_2})
= \theta \cdot (1 - \theta_1)
\]

\(N\) candies, \(r_c\) red-wrapped cherry candies, etc.:

\[
P(X|h_{\theta, \theta_1, \theta_2}) = \theta^c(1 - \theta)\ell \cdot \theta_1^{r_c}(1 - \theta_1)^{g_c} \cdot \theta_2^{r_\ell}(1 - \theta_2)^{g_\ell}
\]

\[
L = [c \log \theta + \ell \log (1 - \theta)]
+ [r_c \log \theta_1 + g_c \log (1 - \theta_1)]
+ [r_\ell \log \theta_2 + g_\ell \log (1 - \theta_2)]
\]
Derivatives of $L$ contain only the relevant parameter:

\[
\frac{\partial L}{\partial \theta} = \frac{c}{\theta} - \frac{\ell}{1-\theta} = 0 \quad \Rightarrow \quad \theta = \frac{c}{c + \ell}
\]

\[
\frac{\partial L}{\partial \theta_1} = \frac{r_c}{\theta_1} - \frac{g_c}{1-\theta_1} = 0 \quad \Rightarrow \quad \theta_1 = \frac{r_c}{r_c + g_c}
\]

\[
\frac{\partial L}{\partial \theta_2} = \frac{r_\ell}{\theta_2} - \frac{g_\ell}{1-\theta_2} = 0 \quad \Rightarrow \quad \theta_2 = \frac{r_\ell}{r_\ell + g_\ell}
\]
Why learn models with hidden variables?

Hidden variables $\Rightarrow$ simplified structure, fewer parameters
$\Rightarrow$ faster learning
Learning with and without hidden variables

![Graph showing learning with and without hidden variables](image-url)
EM for Bayes nets

For $t = 0$ to $\infty$ (until convergence) do

**E step:** Compute all $p_{ijk\ell} = P(X_j = k, Parents(X_j) = \ell \mid e^{(i)}, \theta^{(t)})$

**M step:** $\theta_{jk\ell}^{(t+1)} = \frac{\hat{N}_{jk\ell}}{\sum_{k'} \hat{N}_{jk'\ell}} = \frac{\sum_{i} p_{ijk\ell}}{\sum_{i} \sum_{k'} p_{ijk'\ell}}$

E step can be any exact or approximate inference algorithm

With MCMC, can treat each sample as a complete-data example
Candy example

\begin{itemize}
  \item $P(Bag=1) = \theta$
  \item $P(F=cherry \mid B)$
  \begin{tabular}{|c|c|}
    \hline
    Bag & $P(F=cherry \mid B)$ \tabularnewline
    \hline
    1 & $\theta_{F1}$ \tabularnewline
    2 & $\theta_{F2}$ \tabularnewline
    \hline
  \end{tabular}
\end{itemize}

\begin{itemize}
  \item Flavor
  \item Wrapper
  \item Hole
\end{itemize}

\begin{figure}
\centering
\begin{tikzpicture}
  \node (bag) at (0,0) {Bag};
  \node (flavor) at (-2,-2) {Flavor};
  \node (wrapper) at (0,-2) {Wrapper};
  \node (hole) at (2,-2) {Hole};

  \draw[->] (bag) -- (flavor);
  \draw[->] (bag) -- (wrapper);
  \draw[->] (bag) -- (hole);

  \node at (-2,-1) {$P(Bag=1)$};
  \node at (-2,-1.5) {$\theta$};
\end{tikzpicture}
\end{figure}

\begin{figure}
\centering
\begin{tikzpicture}
  \begin{axis}[
    xlabel={Iteration number},
    ylabel={Log-likelihood $L$},
    xmin=0, xmax=120,
    ymin=-2020, ymax=-1980,
    xtick={0,20,40,60,80,100,120},
    legend pos=north west,
  ]
    \addplot[red] table [x=iteration, y=loglikelihood] {data.csv};
    \addplot[green] table [x=iteration, y=loglikelihood] {data.csv};
  \end{axis}
\end{tikzpicture}
\end{figure}
Example: Car insurance

- Age
- GoodStudent
- ExtraCar
- Mileage
- VehicleYear
- RiskAversion
- SeniorTrain
- MakeModel
- DrivQuality
- Accident
- Antilock
- Airbag
- CarValue
- HomeBase
- AntiTheft
- Theft
- OwnDamage
- OtherCost
- OwnCost
- PropertyCost
- LiabilityCost
- MedicalCost
- Cushioning
- Ruggedness
**Example: Car insurance contd.**

![Graph showing average negative log likelihood per case against number of training cases for different network configurations. The graph includes lines representing different network architectures: 12--3 network, APN algorithm; Insurance network, APN algorithm; 1 hidden node, NN algorithm; 5 hidden nodes, NN algorithm; 10 hidden nodes, NN algorithm; Target network. Each line indicates the performance of the corresponding network architecture across varying numbers of training cases.]
Bayesian learning in Bayes nets

Parameters become variables (parents of their previous owners):

\[ P(\text{Wrapper} = \text{red} \mid \text{Flavor} = \text{cherry}, \Theta_1 = \theta_1, \Theta_2 = \theta_2) = \theta_1. \]

network replicated for each example, parameters shared across all examples:
Bayesian learning contd.

Priors for parameter variables: Beta, Dirichlet, Gamma, Gaussian, etc.

With independent Beta or Dirichlet priors,
MAP EM learning = pseudocounts + expected counts:

\[
M \text{ step: } \theta_{jk\ell}^{(t+1)} = a_{jk\ell} + \frac{\hat{N}_{jk\ell}}{\sum_{k'} \hat{N}_{jk'\ell}}
\]

Implemented in EM training mode for Hugin and other Bayes net packages
Summary (parameter learning)

Complete data: likelihood factorizes; each parameter $\theta_{jkl}$ learned separately from the observed counts for conditional frequency $N_{jkl}/N_{jl}$

Incomplete data: likelihood is a summation over all values of hidden variables; can apply EM by computing “expected counts”

Bayesian learning: parameters become variables in a replicated model with their own prior distributions defined by hyperparameters; then

**Bayesian learning is just ordinary inference in the model**