Introduction to Artificial Intelligence

Lecture 13 – Approximate Inference

CS/CNS/EE 154

Andreas Krause
Bayesian networks

- Compact representation of distributions over large number of variables
- (Often) allows efficient exact inference (computing marginals, etc.)

HailFinder
56 vars
~ 3 states each

⇒~$10^{26}$ terms
> 10,000 years
on Top supercomputers

JavaBayes applet
Typical queries: Conditional distribution

- Compute distribution of some variables given values for others

\[ P(E \mid J = T) \]
\[ P(E, B \mid J = T, M = F) \]
\[ = \frac{1}{2} P(E, B, J = T, M = F) \]
Typical queries: Maximization

- **MPE (Most probable explanation):** Given values for some vars, compute most likely assignment to all remaining vars
  \[
  (a^*, e^*, b^*) = \operatorname{argmax}_{e, b, a} P(E=e, B=b, A=a | J=T, M=F)
  \]

- **MAP (Maximum a posteriori):** Compute most likely assignment to some variables
  \[
  (E^*, B^*) = \operatorname{argmax}_{e, b} P(e, b | J=T, M=F)
  \]

*More general than MPE*
Hardness of inference for general BNs

- Computing conditional distributions:
  - Exact solution: \#P-complete
  - NP-hard to obtain any nontrivial approximation
    \[ \text{E.g. NP-hard to obtain } \hat{p} \text{ s.t. } |P - \hat{p}| < \frac{1}{2} \]

- Maximization:
  - MPE: NP-complete
  - MAP: NP^{PP}-complete

- Inference in general BNs is really hard 😞
Inference

- Can exploit structure (conditional independence) to efficiently perform **exact inference** in many practical situations

- For BNs where exact inference is not possible, can use algorithms for **approximate inference** (later)
Variable elimination algorithm

- Given BN and Query $P(X \mid E=e)$
- Choose an ordering of $X_1, \ldots, X_n$
- Set up initial factors: $f_i = P(X_i \mid Pa_i)$
- For $i = 1:n$, $X_i \notin \{X, E\}$
  - Collect all factors $f$ that include $X_i$
  - Generate new factor by marginalizing out $X_i$
    $$g = \sum_{x_i} \prod_{j} f_j$$
  - Add $g$ to set of factors
- Renormalize $P(x, e)$ to get $P(x \mid e)$
Often, want to compute conditional distributions of many variables, for fixed observations
E.g., probability of *Pits* at different locations given observed *Breezes*
Repeatedly performing variable elimination is wasteful (many factors are recomputed)
Need right data-structure to avoid recomputation
Message passing on factor graphs
Factor graphs

\[ P(C,D,G,I,S,L) = P(C) \cdot P(I) \cdot P(D|C) \cdot P(G|D,I) \cdot P(S|I,G) \cdot P(L|S) \]

- \( f_1(C,D) = P(C) \cdot P(D|C) \)
- \( f_2(D,I,G) = P(I) \cdot P(G|D,I) \)
- \( f_3(I,G,S) = P(S|I,G) \)
- \( f_4(S,L) = P(L|S) \)
A factor graph for a Bayesian network is a bipartite graph consisting of:

- Variables
- Factors

Each factor is associated with a subset of variables, and all CPDs of the Bayesian network have to be assigned to one of the factor nodes.
Sum-product message passing on factor graphs

- Messages from node $v$ to factor $u$
  \[ M_{v \rightarrow u}(x_v) = \prod_{u' \in N(v) \setminus v_j} M_{u' \rightarrow v}(x_v) \]

- Messages from factor $u$ to node $v$
  \[ M_{u \rightarrow v}(x_v) = \sum_{x_u \sim x_v} f_u(x_u) \prod_{u' \in N(u) \setminus u_j} M_{u' \rightarrow u}(x_{u'}) \]
Example messages

\[ \mu_{v \rightarrow u}(x_v) = \prod_{u' \in N(v) \setminus \{u\}} \mu_{u' \rightarrow v}(x_v) \]

\[ \mu_{u \rightarrow v}(x_v) = \sum_{x_u \sim x_v} f_u(x_u) \prod_{v' \in N(u) \setminus \{v\}} \mu_{v' \rightarrow u}(x_{v'}) \]

\[ \mathcal{M}_{A \rightarrow 1}(a) = 1 \]

\[ \mathcal{M}_{1 \rightarrow B}(b) = \sum_a f_1(a, b) \cdot \mathcal{M}_{A \rightarrow 1}(a) = \sum_a \frac{p(a) p(b|a)}{p(a, b)} \cdot 1 = p(b) \]

\[ \mathcal{M}_{B \rightarrow 2}(b) = \mathcal{M}_{1 \rightarrow B}(b) = p(b) \]

\[ \mathcal{M}_{2 \rightarrow C}(c) = \sum_b f_2(b, c) \cdot \mathcal{M}_{B \rightarrow 2}(b) = \sum_b \frac{p(c|b) p(b)}{p(b, c)} = p(c) \]
Belief propagation on polytrees

Belief propagation (aka sum-product) is exact for polytree Bayesian networks

- Factor graph of polytree is a tree
- Choose one node as root
- Send messages from leaves to root, and from root to leaves

After convergence:

\[
P(X_v = x_v) = \prod_{u \in N(v)} \mu_{u \rightarrow v}(x_v)
\]

\[
P(X_u = x_u) = f_u(x_u) \prod_{v \in N(u)} \mu_{v \rightarrow u}(x_v)
\]

Thus: immediately have correct values for all marginals!
What if we have loops?

- Can still apply belief propagation even if we have loops
  - Just run it, close your eyes and hope for the best!
  - Use approximation:
    \[ P(X_v = x_v) \approx \prod_{u \in N(v)} \mu_{u \rightarrow v}(x_v) \]

- In general, will not converge...
- Even if it converges, may converge to incorrect marginals...
- However, in practice often still useful!
  - E.g., turbo-codes, etc.

“Loopy belief propagation”
Behavior of Loopy BP

Loopy BP multiplies same factors multiple times

→ BP often overconfident
Does Loopy BP always converge?

- No! Can oscillate!
- Typically, oscillation the more severe the more “deterministic” the potentials

Graphs from K. Murphy UAI ‘99
What about MPE queries?

E.g.,: What’s the most likely assignment to the unobserved variables, given the observed ones?

Use max-product
(same as sum-product/BP, but with max instead of sums!)

\[
(e^*, b^*, a^*) = \arg \max_{e, b, a} P(e, b, a | j, m) \\
= \arg \max_{e, b, a} P(e, b, a | j, m) \\
= \max_{e, b, a} P(e) P(b) P(a | e, b) P(j | a) P(m | a) \\
= \max_a P(j | a) P(m | a) \max_e P(e) \max_b P(b) P(a | e, b) \\
= \underbrace{g_e(e^*)}_{g(b^*, a^*)}
\]
Max-product message passing on factor graphs

- Messages from nodes to factors

\[ \mu_{v \rightarrow u}(x_v) = \prod_{u' \in N(v) \setminus \{u\}} \mu_{u' \rightarrow v}(x_v) \]

- Messages from factors to nodes

\[ \mu_{u \rightarrow v}(x_v) = \max_{x_u \sim x_v} f_u(x_u) \prod_{v' \in N(u) \setminus \{v\}} \mu_{v' \rightarrow u}(x_{v'}) \]

Replace \( \Sigma \) by \( \max \)
Sampling based inference

- So far: deterministic inference techniques
  - Variable elimination
  - (Loopy) belief propagation

- Will now introduce stochastic approximations
  - Algorithms that “randomize” to compute expectations
  - In contrast to the deterministic methods, guaranteed to converge to right answer (if wait looong enough..)
  - More exact, but slower than deterministic variants
Computing expectations

Often, we’re not necessarily interested in computing marginal distributions, but certain expectations:

- Moments (mean, variance, ...)

\[ \mathbb{E}_P[X^k] = \int x^k P(x) \, dx \]

- Event probabilities

\[ P(X > c) = \mathbb{E}_P[I_{X>c}] = \int [x > c] P(x) \, dx \]

(in general: \[ \mathbb{E}_P[f(X)] = \int P(x) f(x) \, dx \] for continuous
\[ = \sum_k P(x_k) f(x_k) \] for discrete)
Sample approximations of expectations

- $x_1, \ldots, x_N$ samples from RV $X$

  - Law of large numbers:

    $$\mathbb{E}_P[f(X)] = \lim_{N \to \infty} \frac{1}{N} \sum_{i=1}^{N} f(x_i)$$

  - Hereby, the convergence is with probability 1 (almost sure convergence)

  - Finite samples:

    $$\mathbb{E}_p(f(X)) \approx \frac{1}{N} \sum_{i=1}^{N} f(x_i)$$
How many samples do we need?

- **Hoeffding inequality**
  
  Suppose f is bounded in [0,C]. Then
  
  $$P\left(\left|\mathbb{E}_P[f(X)] - \frac{1}{N} \sum_{i=1}^{N} f(x_i)\right| > \varepsilon\right) \leq 2 \exp\left(-\frac{2N\varepsilon^2}{C^2}\right)$$

- Thus, probability of error decreases exponentially in N!

  Suppose I want error at most \(\varepsilon\) with prob. at least 1-\(\delta\)

  $$\exp\left(-\frac{2N\varepsilon^2}{C^2}\right) \leq \frac{\delta}{2}$$

  $$\Rightarrow -\frac{2N\varepsilon^2}{C^2} \leq \ln \frac{\delta}{2}$$

  $$\Rightarrow \frac{2N\varepsilon^2}{C^2} \geq \ln \frac{2}{\delta}$$

  $$\Rightarrow N \geq \frac{1}{2} \frac{C^2}{\varepsilon^2} \ln \frac{2}{\delta}$$

- Need to be able to draw samples from P
Most random number generators produce (approximately) uniformly distributed random numbers.

How can we draw samples from $X \sim \text{Bernoulli}(p)$?

$$Y \sim u([0, 1])$$

$X = 1$ if $Y < p$

$X = 0$ if $Y \geq p$
Sampling from a Multinomial

- $X \sim \text{Mult}([\mu_1, \ldots, \mu_k])$
- where $\mu_i = P(X=i)$; $\sum_i \mu_i = 1$

Function $g$: $[0,1] \rightarrow \{1, \ldots, k\}$ assigns state $g(x)$ to each $x$

Draw sample from uniform distribution on $[0,1]$

Return $g^{-1}(x)$
Forward sampling from a BN

Sample \( D = E, I = L, G = [80, 100] \), \( S = B, L = P \)
Monte Carlo sampling from a BN

- Sort variables in topological ordering $X_1, \ldots, X_n$
- For $i = 1$ to $n$ do
  - Sample $x_i \sim P(X_i \mid X_1=x_1, \ldots, X_{i-1}=x_{i-1})$

- Works even with loopy models!
Computing probabilities through sampling

- Want to estimate probabilities
- Draw N samples from BN
  \[ x_i \ldots x_N \]
- Marginals
  \[
P(L = T) \approx \frac{1}{N} \sum_{i=1}^{N} [L(x_i) = T] = \frac{\text{Count}(L > T)}{N}
  \]
  \= 1 \text{ if } L(x_i) = T
  \= 0 \text{ otherwise}
- Conditionals
  \[
P(L = T | H = F) = \frac{P(L = T, H = F)}{P(H = F)} \approx \frac{\text{Count}(L = T, H = F)}{\text{Count}(H = F)}
  \]
Rejection sampling

- Collect samples over all variables

\[ \hat{P}(X_A = x_A \mid X_B = x_B) \approx \frac{\text{Count}(x_A, x_B)}{\text{Count}(x_B)} \]

- Throw away samples that disagree with \( x_B \)
- Can be problematic if \( P(X_B = x_B) \) is rare event
Sample complexity for probability estimates

- **Absolute error:**

\[
Prob\left(|\hat{P}(x) - P(x)| > \varepsilon\right) \leq
\]

- **Relative error:**

\[
Prob\left(\hat{P}(x) < (1 + \varepsilon)P(x)\right) \leq 2 \exp(-NP(x)\varepsilon^2/3)
\]

If \( P(x) \) exponentially small, need \( N \) exponentially large
Sampling from rare events

- Estimating conditional probabilities $P(X_A \mid X_B = x_B)$ using rejection sampling is hard!
  - The more observations, the unlikelier $P(X_B = x_B)$ becomes

- Want to directly sample from posterior distribution!
Gibbs sampling

- Start with initial assignment $\mathbf{x}^{(0)}$ to all variables.
- For $t = 1$ to $\infty$ do
  - Set $\mathbf{x}^{(t)} = \mathbf{x}^{(t-1)}$
  - For each variable $X_i$
    - Set $\mathbf{v}_i =$ values of all $\mathbf{x}^{(t)}$ except $x_i$
    - Sample $x_i^{(t)}$ from $P(X_i | \mathbf{v}_i)$

- For large enough $t$, sampling distribution will be “close” to true posterior distribution!

**Key challenge**: Computing conditional distributions $P(X_i | \mathbf{v}_i)$