Theoretical Issues in Probabilistic Artificial Intelligence

Michael Kearns

AT&T Labs

Part I: Graphical Models and Probabilistic Inference

Representing Distributions by Directed Graphs

- Joint distribution $P(X_1, \ldots, X_n)$ on boolean variables
- Conditional factorization:

$$\begin{aligned} P(X_1, \ldots, X_n) &= P(X_1)P(X_2|X_1) \cdots P(X_n|X_1, \ldots, X_{n-1}) \\ &= \prod P(X_i|X_1, \ldots, X_{i-1}) \end{aligned}$$

- Hope for simplifications through conditional independences:

$$P(X_5|X_1, \ldots, X_4) = P(X_5|X_3)$$

Example: Burglar Alarm Model

- Variables A(larm), B(urglar), E(arthquake), J(ohn), M(ary)
- Joint distribution $P(A, B, E, J, M)$
- Exploit **causality** to choose ordering
- **Assert** factorization

$$P(A, B, E, J, M) = P(B)P(E)P(A|B, E)P(J|A)P(M|A)$$

- Associated **directed graph**: if factorization contains $P(X|pa(X))$, have directed edges from $pa(X)$ to X
- No directed loops, but may have undirected loops
- Full model = directed graph (factorization) + CPT's

Advantages of Bayes Nets

- Dimensionality reduction: $O(2^n) \to O(2^k n)$ parameters, k = max in-degree ($31 \to 10$ parameters in burglar alarm)
- Separate **causality** (qualitative) from CPT's (quantitative)
- Hidden variables can simplify model
- Graph-theoretic algorithms for natural problems

Caveats About Bayes Nets

- **Order** of decomposition can be crucial
- Generally want to reduce in-degree, undirected loops
- Basic problems still notoriously hard;
  must find special cases of interest

Basic Problems on Bayes Nets

- **Inference:**
  - Set S of instantitated **evidence variables** (e.g., $S = \{X_2 = 0, X_7 = 1\}$)
  - **Query variable(s)** X
  - **Goal**: compute $P(X|S)$
  - Query types: diagnostic, predictive,. . .
- **Learning:**
  - **Parameter estimation**: given directed graph; must learn CPT's from sample data
  - **Structure learning**: learn directed graph (and CPT's) from sample data

Complexity of the Basic Problems

- **Inference:**
  - $\#P$-complete in the worst case;
    many intractable restrictions
  - Interesting algorithms for several special cases
- **Learning:**
  - Efficient parameter estimation from fully observed data,
    good heuristics for partially observable
  - Structure learning: intractable

Subtleties of Conditional Independence: “Explaining Away”

- Two variables that are **independent** with **no** evidence may become **dependent** in the presence of evidence
- Burglar alarm example: B and E are independent, but if we observe $A = 1$ then they are dependent
- If we learn there was an earthquake, less likely to believe there was a burglary
- What determines when X and Y are independent given S?

A Graph-Theoretic Characterization of Independence: d-Separation

Let P be an undirected path between X and Y. Say that P is **blocked** by S if:

- There is a node $Z \in S$ on P with an out-edge along P;
- There is a node $Z \notin S$ on P, with both edges along P directed in, and no descendant of Z is in S.

All paths blocked: d-separation, and

$$P(X, Y|S) = P(X|S)P(Y|S)$$

A Tractable Special Case for Inference: Polytrees

- **Polytree**: no **undirected** cycles
- Query node X, evidence set S, want to compute $P(X = x|S)$
- Let $S(X,Y) \subseteq S$ be the evidence **reachable** (undirected) from X **avoiding** Y
- Algorithm: for all nodes X, Y, if $X \to Y$:
  - X sends to Y: $P(X = x, S(X,Y))$ for each x
  - Y sends to X: $P(S(Y,X)|X = x)$ for each x
- S^+, S^-: evidence "upstream" and "downstream" from query node X

Analysis

- First write $P(X=x|S) = P(X=x,S)/P(S) = \alpha P(X=x,S)$
- By d-separation on X:

$$
\begin{aligned}
P(X=x,S) &= P(X=x,S^{+})P(S^{-}|X=x,S^{+}) \\
&= P(X=x,S^{+})P(S^{-}|X=x)
\end{aligned}
$$

- Compute $P(X=x,S^{+})$, $P(S^{-}|X=x)$ from messages to X
- Let $\vec{U}$ be parents of X, $\vec{V}$ be children of X

Analysis Continued...

- Computing $P(X = x, S^+)$: marginalize over parents

$$P(X = x, S^+) = \sum_{\vec{u}} P(\vec{U} = \vec{u}, S^+) P(X = x | \vec{U} = \vec{u}, S^+)$$

- $P(X = x | \vec{U} = \vec{u}, S^+) = P(X = x | \vec{U} = \vec{u})$, get from CPT
- $P(\vec{U} = \vec{u}, S^+) = \prod P(U_i = u_i, S(U_i, X))$ by d-separation on X; messages from U_i to X
- $P(S^- | X = x) = \prod P(S(V_i, X) | X = x)$ by d-separation on X
- Messages from children V_i to X

Wrapping Up

- If X has all but message from Y, can write to Y
- Tree fills up from the leaves
- Running time: linear in tree size and CPT size

Generalizations to Sparse Networks

Two basic approaches:

- **Cluster** nodes until a polytree is obtained
- **Instantiate** some nodes to yield a set of polytrees, take weighted average

Run time typically exponential in cluster size or number of instantiated variables.

Approximate Inference in Dense Networks

- Often assume a **parametric form** for CPT's
- Parametric form assures "randomness" or averaging behavior
- Sampling/simulation methods: Gibbs sampling
- Variational methods: rigorous upper and lower bounds

Some Common Parametric CPT's

- Node X, parents $U_1, \ldots, U_n$
- CPT specified by weight vector $\vec{\theta}$
- Look at forms Pr$[X = 1|\vec{U} = \vec{u}] = \sigma(\vec{\theta} \cdot \vec{u})$
- **Noisy-OR**: $\sigma(x) = 1 - e^{-x}$
- **Sigmoid**: $\sigma(x) = 1/(1 + e^{x})$

Inference in Two-Layer Noisy-OR Networks

- Input units $U_1, \ldots, U_n$, outputs $X_1, \ldots, X_m$
- CPT's for outputs given by weight vectors $\vec{\theta}^1, \ldots, \vec{\theta}^m$
- Inputs have biases $p_1, \ldots, p_n$, assume all 1/2
- Can reduce general queries to form

$$\Pr[X_1 = 1, \ldots, X_m = 1] = (1/2^n) \sum_{\vec{u}} \left(\prod_i \left(1 - e^{-\vec{\theta}^i \cdot \vec{u}} \right) \right)$$

- Suppose we choose λ_i, $i = 1, \ldots, m$, such that

$$e^{\lambda_i x} \geq 1 - e^{-x}$$

for **all** x

Closed-Form Computation of the Variational Upper Bound

$$
\begin{aligned}
(1/2^n)\sum_{\vec{u}}\prod_i\left(e^{\lambda_i\vec{\theta^i}\cdot\vec{u}}\right) &= (1/2^n)\sum_{\vec{u}}\left(e^{\sum_i\lambda_i\vec{\theta^i}\cdot\vec{u}}\right)\\
&= (1/2^n)\sum_{\vec{u}}\left(e^{\sum_i\lambda_i\sum_j\theta^i_j u_j}\right)\\
&= (1/2^n)\sum_{\vec{u}}\left(e^{\sum_j u_j\sum_i\lambda_i\theta^i_j}\right)\\
&= (1/2^n)\sum_{\vec{u}}\left(\prod_j\left(e^{u_j\sum_i\lambda_i\theta^i_j}\right)\right)\\
&= \prod_j\mathbf{E}\left[e^{u_j\sum_i\lambda_i\theta^i_j}\right]
\end{aligned}
$$

How Should We Choose the λ_i?

- Basic idea: over the distribution on the weighted sums, integrate an **upper bound** on transfer function
- Single unit: choose λ_i so upper bound approximates transfer function well near $\mu_i = \mathbf{E}[\vec{\theta^i} \cdot \vec{u}]$
- Many units: may do better than approximating near each μ_i
- The λ_i capture (limited) correlations between the X_i
- In practice: gradient descent on $\vec{\lambda}$

Analysis of Variational Methods

- Let P be true probability, $\widehat{P}_U(\vec{\lambda})$, $\widehat{P}_L(\vec{\lambda})$ variational upper and lower bounds
- Want to bound $\widehat{P}_U(\vec{\lambda}) - \widehat{P}_L(\vec{\lambda})$
- Intuition: for "most" input settings $\vec{u}$, all weighted sums are "near" their means

Large Deviation Methods

- Probability that $\vec{\theta}^i \cdot \vec{u}$ exceeds its mean μ_i by more than ϵ_i bounded by $e^{c_i \epsilon_i^2 n}$
- **Conditioned** on this event E_i, $\Pr[X_i = 1 | E_i] \leq \sigma(\mu_i + \epsilon_i)$
- Probability **some** E_i fails bounded by $\sum_i e^{c_i \epsilon_i^2 n}$
- Another parameterized upper bound:

$$\hat{P}_U(\vec{\epsilon}) = \left(1 - \sum_i e^{c_i \epsilon_i^2 n}\right) \prod_i \sigma(\mu_i + \epsilon_i) + \sum_i e^{c_i \epsilon_i^2 n}$$

- Lower bound:

$$\hat{P}_L(\vec{\epsilon}) = \left(1 - \sum_i e^{c_i \epsilon_i^2 n}\right) \prod_i \sigma(\mu_i - \epsilon_i)$$

Bounds for Large, Dense Networks

- Can get bounds of form

$$m/n^2 + \beta^m m\sqrt{\log(n)/n}$$

  for some $\beta < 1$ depending on network

- Larger γ yields larger ϵ_i

- Generalizes to variational methods, arbitrary transfer functions, multilayer networks,...

Further Topics

- Handling "loopy" networks: connections with decoding turbocodes
- Object-oriented Bayesian networks