Bayesian Networks: Inference

Inference in BN

Simplest case:

For Boolean variables:

\[ P(b) = \sum_A P(A, b) \]
\[ = \sum_A P(B|A)P(A) \]

For Boolean variables:

\[ P(b) = P(b|a)P(a) + P(b|\neg a) P(\neg a) \]
\[ P(\neg b) = P(\neg b|a)P(a) + P(\neg b|\neg a) P(\neg a) \]

Inference on a chain

Summary:

\[ P(D) = \sum_{A, B, C, D} P(A, B, C, D) \]
\[ = \sum_{A, B, C} P(D|C)P(C|B)P(B|A)P(A) \]
\[ = \sum_C P(D|C) \sum_B P(C|B) \sum_A P(B|A)P(A) \]
Inference on a chain

We can perform variable elimination using a different order:

\[
P(D) = \sum_{A,B,C} P(A)P(B|A)P(C|B)P(D|C)
\]

\[
= \sum_{A} P(A) \sum_{B} P(B|A) \sum_{C} P(D|C)P(B|C)
\]

\[
= \sum_{A} P(A) \sum_{B} P(B|A) f_c(B, D)
\]

What is the difference in efficiency?

Variable Elimination

- Suppose we're interested in \( P(X_k) \)
- Write query in the form
  \[
  P(X_k) = \sum_{X_\text{other}} \prod P(X_i|P_a(X_i))
  \]
- Iteratively
  - Move all irrelevant terms outside of innermost sum
  - Perform innermost sum, getting a new term
  - Insert the new term into the product

Inference Example 2

\[
P(W) = \sum_{R,S} P(W|R,S)P(R|C)P(S|C)P(C)
\]

\[
= \sum_{R,S} P(W|R,S) f_c(r,s) f_c(r,s)
\]

A More Complex Example

The “Asia” network:

Example (cont)

Suppose we want to compute \( P(D) \)

The form of the joint distribution:

\[
P(v)p(s)p(t|v)p(l|s)p(b|s)p(a|t,l)p(x|a)p(d|a,b)
\]

Need to eliminate: \( v,s,x,t,l,a,b \)

\[
= f_c(t) = \sum_{v} P(v)p(t|v)
\]

Compute:

\[
f_c(t) = f_c(t)p(s)p(l|s)p(b|s)p(a|t,l)p(x|a)p(d|a,b)
\]

Note: \( f_c(t) \neq P(t) \)

In general, result of elimination is not necessarily a probability term
Example (cont)

Need to eliminate: $s, t, l, a, b$

\[
P(v)P(s|v)P(l|s)P(b|s)P(a|t,l)P(x|a)P(d|a, b)
\]

\[
\Rightarrow f(t)P(s|v)P(l|s)P(b|s)P(a|t,l)P(x|a)P(d|a, b)
\]

Eliminate: $s$

Compute:

\[
f_f(b,l) = \sum f_s(y)P(b|s)P(l|s)
\]

\[
\Rightarrow f_f(b,l)P(a|t,l)P(x|a)P(d|a, b)
\]

Summing on $s$ results in a factor with two arguments $f_f(b,l)$. In general, result of elimination may be a function of several variables.

Example (cont)

Need to eliminate: $t, l, a, b$

\[
P(v)P(s|v)P(l|s)P(b|s)P(a|t,l)P(x|a)P(d|a, b)
\]

\[
\Rightarrow f_f(t)P(s|v)P(l|s)P(b|s)P(a|t,l)P(x|a)P(d|a, b)
\]

\[
\Rightarrow f_f(t)f_f(b,l)P(a|t,l)P(x|a)P(d|a, b)
\]

Eliminate: $t$

Compute:

\[
f_f(a,l) = \sum f_f(t)P(a|t,l)
\]

\[
\Rightarrow f_f(b,l)f_f(a,l)P(d|a, b)
\]

Example (cont)

Need to eliminate: $l, a, b$

\[
P(v)P(s|v)P(l|s)P(b|s)P(a|t,l)P(x|a)P(d|a, b)
\]

\[
\Rightarrow f_f(t)P(s|v)P(l|s)P(b|s)P(a|t,l)P(x|a)P(d|a, b)
\]

\[
\Rightarrow f_f(t)f_f(b,l)P(a|t,l)P(x|a)P(d|a, b)
\]

\[
\Rightarrow f_f(t)f_f(b,l)f_f(a,l)P(d|a, b)
\]

Eliminate: $l$

Compute:

\[
f_f(a,b) = \sum f_f(b,l)f_f(a,l)
\]

\[
\Rightarrow f_f(a,b)f_f(a,l)P(d|a, b)
\]

Example (cont)

Need to eliminate: $a, b$

\[
P(v)P(s|v)P(l|s)P(b|s)P(a|t,l)P(x|a)P(d|a, b)
\]

\[
\Rightarrow f_f(t)P(s|v)P(l|s)P(b|s)P(a|t,l)P(x|a)P(d|a, b)
\]

\[
\Rightarrow f_f(t)f_f(b,l)P(a|t,l)P(x|a)P(d|a, b)
\]

\[
\Rightarrow f_f(t)f_f(b,l)f_f(a,l)P(d|a, b)
\]

\[
\Rightarrow f_f(t)f_f(b,l)f_f(a,l)f_f(0,a)P(d|a, b)\Rightarrow f_f(d)
\]

Eliminate: $a, b$

Compute:

\[
f_f(b,d) - \sum f_f(a,b)f_f(a)P(d|a, b)
\]

\[
f_f(d) - \sum f_f(b,d)
\]

Variable Elimination

- We now understand variable elimination as a sequence of rewriting operations.
- Computation depends on order of elimination.
Dealing with Evidence

How do we deal with evidence?
- Suppose get evidence \( V = t, S = f, D = t \) and want to compute \( P(L | V = t, S = f, D = t) \)

\[
P(L | V = t, S = f, D = t) = \frac{P(L, V = t, S = f, D = t)}{P(V = t, S = f, D = t)}
\]

Eliminating initial factors, after setting evidence:
- We start by writing the factors:
  \[
P(v)P(s|v)p(t|v)p(l|s)p(b|s)p(a|t,l)p(x|a)p(d|a,b)
\]
- Since we know that \( V = t \), we don’t need to eliminate \( V \)
- Instead, we can replace the factors \( P(V) \) and \( P(T/V) \) with
  \[
f_{p(v)} = P(V = t) \quad f_{p(t|v)} = P(T | V = t)
\]
- These “select” the appropriate parts of the original factors given the evidence
- Note that \( f_{p(v)} \) is a constant, and thus does not appear in elimination of other variables

---

Dealing with Evidence

Compute \( P(L, V = t, S = f, D = t) \)
- Initial factors, after setting evidence:
  \[
f_{p(v)}f_{p(t|v)}f_{p(l|s)}f_{p(b|s)}f_{p(a|t,l)}f_{p(x|a)}f_{p(d|a,b)}
\]

Eliminating \( X \):
- We get
  \[
f_{p(v)}f_{p(t|v)}f_{p(l|s)}f_{p(b|s)}f_{p(a|t,l)}f_{p(x|a)}f_{p(d|a,b)}
\]

---

Variable Elimination

- Compute the probability of \( X_l \) given values to evidence variables \( E \)
  \[
P(X_l, E) = \sum_{\text{non-query, non-evidence variables}} \prod P(X_i | P(E(X_i)))
\]
- Algorithm is same as before, with no need to perform summation with respect to evidence variables.
Complexity of variable elimination

- Suppose in one elimination step we compute
  \[ f_k(y_1, ..., y_k) = \sum f_{i+1}(x_i, y_1, ..., y_k) \]
  \[ f_{i+1}(x_i, y_1, ..., y_k) = \prod f(x_i, y_{1i}, ..., y_{ki}) \]

  This requires
  \[ m \cdot \text{Val}(X) \cdot \prod \text{Val}(Y) \] multiplications
  - For each value for \( x, y \), we do \( m \) multiplications

  \[ \text{Val}(X) \cdot \prod \text{Val}(Y) \] additions
  - For each value of \( y \), we do \( \text{Val}(X) \) additions

  Complexity is exponential in the number of variables in the intermediate factor. Finding an ordering which leads to a minimal number of variables in factors is NP-hard!

Complexity of inference

Thm:
Computing \( P(X = x) \) in a Bayesian network is NP-hard

Proof

Reduce 3-SAT to Bayesian network inference

We are given a 3-SAT problem:
- \( q_1, ..., q_n \) be propositional symbols,
- \( \Psi_1, ..., \Psi_k \) be clauses, such that \( \Psi_i = l_{i1} \lor l_{i2} \lor l_{i3} \) where each \( l_{ij} \) is a literal over \( q_1, ..., q_n \)
- \( \Psi = \Psi_1 \land ... \land \Psi_k \)

We will construct a network s.t. \( P(X = t) > 0 \) iff \( \Psi \) is satisfiable

Proof (cont)

- It is easy to check
  - Polynomial number of variables
  - Each CPDs can be described by a small table (8 parameters at most)
  - \( P(X = \text{true}) > 0 \) if and only if there exists a satisfying assignment to \( Q_1, ..., Q_n \)

- Conclusion: polynomial reduction of 3-SAT to BN inference

The network

- \( P(Q_i = \text{true}) = 0.5 \)
- \( P(\Psi = \text{true} \mid Q_i, Q_j, Q_l) = 1 \) iff \( Q_i, Q_j, Q_l \) satisfy the clause \( \Psi \)
- \( A_1, A_2, ... \) are binary OR gates

It’s even worse than we thought…

- This construction shows a stronger result.
- \( \#P \) - the complexity class associated with counting problems.

- In our reduction, \( 2^P(X = t) \) is the number of satisfying assignments to \( \Psi \)
- Thus, BN inference is \( \#P \)-hard (in fact, \( \#P \)-complete).
Exercise: Variable elimination

Query: What is the probability that a student is smart, given that they pass the exam?

Approaches to inference

- **Exact inference**
  - Variable elimination
  - Join tree algorithm
- **Approximate inference**
  - Simplify the structure of the network to make exact inference efficient (variational methods, loopy belief propagation)
- **Probabilistic methods**
  - Stochastic simulation / sampling methods
  - Markov chain Monte Carlo methods