Probabilities in BNs

- For all joint distributions, we have (chain rule):
  \[ P(x_1, x_2, \ldots, x_n) = \prod_{i=1}^{n} P(x_i | x_1, \ldots, x_{i-1}) \]

- Bayes’ nets implicitly encode joint distributions
  - As a product of local conditional distributions
  - To see what probability a BN gives to a full assignment, multiply all the relevant conditionals together:
    \[ P(x_1, x_2, \ldots, x_n) = \prod_{i=1}^{n} P(x_i | \text{parents}(x_i)) \]

- This lets us reconstruct any entry of the full joint
  - Not every BN can represent every joint distribution
    - The topology enforces certain conditional independencies

All Conditional Independences

- Given a Bayes net structure, can run d-separation to build a complete list of conditional independences that are necessarily true of the form
  \[ X_i \perp X_j \mid \{X_{k_1}, \ldots, X_{k_n}\} \]

- This list determines the set of probability distributions that can be represented

Same Assumptions, Different Graphs?

- Can you have two different graphs that encode the same assumptions?
  - Yes!
  - Examples:

Example: Independence

- For this graph, you can fiddle with \( \theta \) (the CPTs) all you want, but you won’t be able to represent any distribution in which the flips are dependent!
Topology Limits Distributions

- Given some graph topology $G$, only certain joint distributions can be encoded.
- The graph structure guarantees certain (conditional) independences.
- (There might be more independence).
- Adding arcs increases the set of distributions, but has several costs.
- Full conditioning can encode any distribution.

Causality?

- When Bayes’ nets reflect the true causal patterns:
  - Often simpler (nodes have fewer parents).
  - Often easier to think about.
  - Often easier to elicit from experts.
- BNs need not actually be causal.
  - Sometimes no causal net exists over the domain.
  - E.g., consider the variables Traffic and Drips.
  - End up with arrows that reflect correlation, not causation.
- What do the arrows really mean?
  - Topology may happen to encode causal structure.
  - Topology only guaranteed to encode conditional independence.
- *More about causality: [Causality – Judea Pearl]*

Bayes Nets Representation Summary

- Bayes nets compactly encode joint distributions.
- Guaranteed independencies of distributions can be deduced from BN graph structure.
- D-separation gives precise conditional independence guarantees from graph alone.
- A Bayes’ net’s joint distribution may have further (conditional) independence that is not detectable until you inspect its specific distribution.

Inference

- Inference: calculating some useful quantity from a joint probability distribution.
- Examples:
  - Posterior probability:
    $$P(Q|E_1 = e_1, \ldots, E_k = e_k)$$
  - Most likely explanation:
    $$\text{argmax}_Q P(Q = q|E_1 = e_1, \ldots)$$

Inference by Enumeration

- Given unlimited time, inference in BNs is easy.
- Recipe:
  - State the marginal probabilities you need.
  - Figure out ALL the atomic probabilities you need.
  - Calculate and combine them.
- Example:
  $$P(+b, +j, +m) = \frac{P(+b, +j, +m) \cdot P(+m|+a)}{P(+j, +m)}$$

Example: Enumeration

- In this simple method, we only need the BN to synthesize the joint entries.
  $$P(+b, +j, +m) = P(+b)P(+j)P(+a)P(+b|+a)P(+j|+a)P(+m|+a) + P(+b)P(+j)P(+a|+b, +e)P(+j|a)P(+m|+a) + P(+b)P(+j)P(+a|+b, -e)P(+j|a)P(+m|+a) + P(+b)P(+j)P(+a|+b, -e)P(+j|a)P(+m|-a) + P(+b)P(+j)P(+a|+b, -e)P(+j|a)P(+m|-a)$$
Inference by Enumeration?

Why is inference by enumeration so slow?
- You join up the whole joint distribution before you sum out the hidden variables
- You end up repeating a lot of work!

Idea: interleave joining and marginalizing!
- Called “Variable Elimination”
- Still NP-hard, but usually much faster than inference by enumeration
- We’ll need some new notation to define VE

Factor Zoo I

Joint distribution: $P(X,Y)$
- Entries $P(x,y)$ for all $x, y$
- Sums to 1

Selected joint: $P(x,Y)$
- A slice of the joint distribution
- Entries $P(x,y)$ for fixed $x$, all $y$
- Sums to $P(x)$

Factor Zoo II

Family of conditionals: $P(X|Y)$
- Multiple conditionals
- Entries $P(x | y)$ for all $x, y$
- Sums to $|Y|

Single conditional: $P(Y | x)$
- Entries $P(y | x)$ for fixed $x$, all $y$
- Sums to 1

Factor Zoo III

Specified family: $P(y | X)$
- Entries $P(y | x)$ for fixed $y$, but for all $x$
- Sums to … who knows!

In general, when we write $P(Y_1 … Y_n | X_1 … X_m)$
- It is a “factor,” a multi-dimensional array
- Its values are all $P(y_1 … y_n | x_1 … x_m)$
- Any assigned $X$ or $Y$ is a dimension missing (selected) from the array

Example: Traffic Domain

Random Variables
- R: Raining
- T: Traffic
- L: Late for class!

First query: $P(L)$

$P(T|R)$

$P(R)$

$P(L|R)$
Track objects called factors
- Initial factors are local CPTs (one per node)
  \[ P(R), P(T|R), P(L|T) \]
  - Any known values are selected
    - E.g. if we know \( L = +\ell \), the initial factors are
  \[ P(R), P(T|R), P(+\ell|T) \]
  - VE: Alternately join factors and eliminate variables

Example: Multiple Joins

Operation 1: Join Factors
- First basic operation: joining factors
- Combining factors:
  - Just like a database join
  - Get all factors over the joining variable
  - Build a new factor over the union of the variables involved
- Example: Join on \( R \)
  \[ P(R) \times P(T|R) \rightarrow P(R, T) \]
  - Computation for each entry: pointwise products
    \[ \Pr_{r,t} : \ P(r,t) = P(r) \cdot P(t|r) \]

Operation 2: Eliminate
- Second basic operation: marginalization
- Take a factor and sum out a variable
  - Shrinks a factor to a smaller one
  - A projection operation
- Example:
  \[ P(R, T) \rightarrow P(T) \]
Marginalizing Early (aka VE*)

- **Evidence**
  - If evidence, start with factors that select that evidence
    - No evidence uses these initial factors:
      
      \[
      \begin{align*}
      P(R) & : P(R) = \begin{cases} 0.1 & \text{if } r \\ 0.9 & \text{if } \neg r \end{cases} \\
      P(T|R) & : P(T|R) = \begin{cases} 0.8 & \text{if } t, r \\ 0.2 & \text{if } t, \neg r \\ 0.2 & \text{if } \neg t, r \\ 0.8 & \text{if } \neg t, \neg r \end{cases} \\
      P(L|R) & : P(L|R) = \begin{cases} 0.3 & \text{if } l, r \\ 0.7 & \text{if } l, \neg r \\ 0.7 & \text{if } \neg l, r \\ 0.3 & \text{if } \neg l, \neg r \end{cases} \\
      P(L|T) & : P(L|T) = \begin{cases} 0.2 & \text{if } l, t \\ 0.8 & \text{if } \neg l, t \\ 0.8 & \text{if } l, \neg t \\ 0.2 & \text{if } \neg l, \neg t \end{cases} \\
      P(T) & : P(T) = \frac{0.17}{0.17 + 0.09 + 0.08 + 0.03} = 0.3 \\
      P(L) & : P(L) = \frac{0.7}{0.7 + 0.7 + 0.7 + 0.7} = 0.7
      \end{align*}
      \]

  - Computing \( P(L | +r) \), the initial factors become:
    
    \[
    \begin{align*}
    P(+r) & : P(+r) = \begin{cases} 0.1 & \text{if } r \\ 0.9 & \text{if } \neg r \end{cases} \\
    P(T|+r) & : P(T|+r) = \begin{cases} 0.8 & \text{if } t, r \\ 0.2 & \text{if } t, \neg r \end{cases} \\
    P(L|+r) & : P(L|+r) = \begin{cases} 0.3 & \text{if } l, r \\ 0.7 & \text{if } l, \neg r \end{cases} \\
    \end{align*}
    \]

  - We eliminate all vars other than query + evidence

- **Evidence II**
  - Result will be a selected joint of query and evidence
    - E.g. for \( P(L | +r) \), we’d end up with:
      
      \[
      \begin{align*}
      P(+r, L) & : P(+r, L) = \frac{0.026}{0.026 + 0.074} = 0.26 \\
      P(L | +r) & : P(L | +r) = \frac{0.074}{0.026 + 0.074} = 0.74
      \end{align*}
      \]

  - To get our answer, just normalize this!
  - That’s it!

General Variable Elimination

- **Query**: \( P(Q|E_1 = e_1, \ldots, E_k = e_k) \)
- **Start with initial factors**:
  - Local CPTs (but instantiated by evidence)
- **While there are still hidden variables (not Q or evidence)**:
  - Pick a hidden variable H
  - Join all factors mentioning H
  - Eliminate (sum out) H
- **Join all remaining factors and normalize**

Variable Elimination Bayes Rule

- **Start / Select Join on B Normalize**
  - **P(B)**
    
    \[
    \begin{align*}
    a & : P(a) = \begin{cases} 0.1 & \text{if } a \\ 0.9 & \text{if } \neg a \end{cases} \\
    \end{align*}
    \]

  - **P(B|a)**
    
    \[
    \begin{align*}
    a & : P(a|b) = \begin{cases} 0.1 & \text{if } a, b \\ 0.9 & \text{if } a, \neg b \end{cases} \\
    \end{align*}
    \]

  - **P(a|B)**
    
    \[
    \begin{align*}
    B & : P(b | a) = \begin{cases} 0.9 & \text{if } b, a \\ 0.1 & \text{if } b, \neg a \end{cases} \\
    a & : P(a | b) = \begin{cases} 0.1 & \text{if } a, b \\ 0.9 & \text{if } a, \neg b \end{cases} \\
    \end{align*}
    \]

  - **P(B|a)**
    
    \[
    \begin{align*}
    B & : P(b | a) = \begin{cases} 0.1 & \text{if } b, a \\ 0.9 & \text{if } b, \neg a \end{cases} \\
    a & : P(a | b) = \begin{cases} 0.1 & \text{if } a, b \\ 0.9 & \text{if } a, \neg b \end{cases} \\
    \end{align*}
    \]
### Variable Elimination

- **What you need to know:**
  - Should be able to run it on small examples, understand the factor creation/reduction flow
  - Better than enumeration: saves time by marginalizing variables as soon as possible rather than at the end

- **We will see special cases of VE later**
  - On tree-structured graphs, variable elimination runs in polynomial time, like tree-structured CSPs
  - You’ll have to implement a tree-structured special case to track invisible ghosts (Project 4)