CS 337, Fall 2023
Bayes Networks and Variable Elimination

Scribes: Martha Nybroe, Khushi Gondane, Hanvitha, Ayush Manaswita*, Hari Prakash*, Pratham Garg*, Ramswaroop Rathi *
Edited by: Vedang Asgaonkar

November 7, 2023

Disclaimer. Please note this document has not received the usual scrutiny that formal publications enjoy. This may be distributed outside this class only with the permission of the instructor.

1 Bayes Networks and Variable Elimination

1.1 Bayes Network

1.1.1 Recall of D-separation

If there are two random variables are they conditionally independent?
Is $X_i \perp X_j \mid \{X_{k1}, \ldots, X_{kn}\}$?

- Check all (undirected) paths between X_i and X_j, with keeping $X_{k1}, \ldots, X_{kn}$ observed.
- If all triples in a path are active, then the path is active. If we find at least one active path across all paths, then the independence does not hold. (Note that even a single inactive triple in a path will render the path inactive.)
- Otherwise, iff **ALL** paths are inactive, then independence is guaranteed, that is X_i and X_j are d-separated given $X_{k1}, \ldots, X_{kn}$.

Using d-separation we can identify all the conditional independencies

1.1.2 Inference

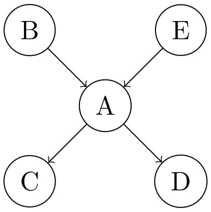
Inference: Given an evidence variable (a set of variables E), what is the probability of a query variable (or a set of variables)

$$P(Q|e_1, \ldots, e_k) = ?$$

Example of inference via naive enumeration:

$$\begin{aligned} P(B|C,D) &\propto P(B,C,D) \\ &\propto P(B,a,c,d,e) \\ &\propto \sum_{e,a} P(B) \cdot P(e) \cdot P(a|B,e) \cdot P(c|a) \cdot P(d|a) \end{aligned}$$

[0] Larger credit goes to these scribes towards the final notes.



Computing joint probability over all random variables and subsequent normalization. This involves aggregating the probabilities associated with each random variable, which can be computationally intensive due to the exponential growth in combinations as the number of variables increases.

To mitigate these computational challenges, several optimization techniques can be employed:

- **Dynamic Programming:** This method efficiently breaks down problems into smaller subproblems, often caching intermediate results to reduce redundant computations and enhance overall efficiency.
- **Factorization Techniques:** Utilizing methods like variable elimination or factor graphs can simplify joint probability computation by representing the probabilities in a more manageable form, reducing unnecessary variables and streamlining calculations.
- **Belief Propagation:** Particularly effective in graphical models, belief propagation leverages the graph structure to distribute information, minimizing redundant calculations and accelerating the derivation of joint probabilities.

By employing these optimized algorithms, the computational complexity of joint probability calculations can be significantly reduced, providing faster and more efficient means to derive complex probabilities in various probabilistic models.

1.2 Variable elimination

Variable Elimination is an **Exact inference** algorithm for Bayes Networks. VE works by eliminating all variables in turn until there is a factor with only query variable. Given an order, this sequence of alternating between joining and marginalizing is a more efficient strategy than naive enumeration. Ordering is critical to get optimal efficiency. Finding optimal ordering is a NP hard problem.

1.2.1 Variable elimination algorithm

Variable Elimination algorithm deals with ”factors”. Factors are a generalized version of CPDs (Conditional Probability Distribution).

Steps for VE algorithm:

1. Instantiate all observed variables with the observed values.
2. Given an order of elimination across the *non-query variables sum out or marginalize one random variable at a time.

*the variables over which we want to sum up are non-query variables.

1.2.2 Example

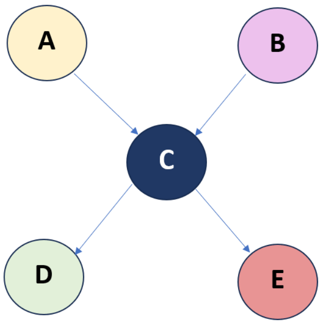


Figure 1: VE example

An illustration of factors:
Query variable: D
Keeping the order of elimination: A-B-E-C

$$\begin{aligned}
P(D) &= \sum_{E,C,B,A} P(D,E,C,B,A) \\
&= \sum_{E,C,B,A} P(D|C)P(E|C)P(B)P(C|B,A)P(A) \\
&= \sum_{C} P(D|C) \sum_{E} P(E|C) \sum_{B} P(B) \sum_{A} P(C|B,A)P(A) \\
&= \sum_{C} P(D|C) \sum_{E} P(E|C) \sum_{B} P(B) f_1(C,B) \\
&= \sum_{C} P(D|C) \sum_{E} P(E|C) f_2(C) \\
&= \sum_{C} P(D|C) f_3(C) \\
&= f_4(D)
\end{aligned}$$

Here f_1, f_2, f_3, f_4 are factors

1.2.3 Order of Elimination

Order of elimination across the non-query variables is important because, it can essentially make only as efficient as naive elimination or significantly more efficient. Changing the order slightly can lead to large changes in efficiency

Consider a naive Bayes model, where we have a class and some variables , which are all conditionally independent given the class

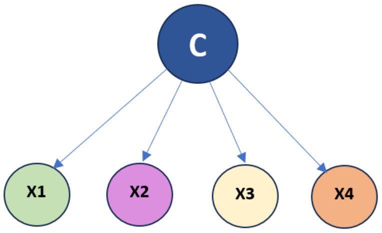


Figure 2: VE Example2

Query variable: X4

Considering order of elimination: X3-X2-X1-C

$$\begin{aligned}
P(X4) &= \sum_{C,X1,X2,X3} P(C)P(X1|C)P(X2|C)P(X3|C)P(X4|C) \\
&= \sum_{C} P(C) \sum_{X1} P(X1|C) \sum_{X2} P(X2|C) \sum_{X3} P(X3|C)P(X4|C) \\
&= \sum_{C} P(C) \sum_{X1} P(X1|C) \sum_{X2} P(X2|C) f_1(X4, C) \\
&= \sum_{C} P(C) \sum_{X1} P(X1|C) f_2(X4, C) \\
&= \sum_{C} P(C) f_3(X4, C)
\end{aligned}$$

Largest factor will involve two random variables i.e X4, C
Now we will slightly change the order as follows

Considering order of elimination: C-X3-X2-X1

$$\begin{aligned}
P(X4) &= \sum_{C,X1,X2,X3} P(C)P(X1|C)P(X2|C)P(X3|C)P(X4|C) \\
&= \sum_{X1} \sum_{X2} \sum_{X3} \sum_{C} P(X1|C)P(X2|C)P(X3|C)P(X4|C) \\
&= \sum_{X1} \sum_{X2} \sum_{X3} f_1(X1, X2, X3, X4)
\end{aligned}$$

Here by eliminating C first , the largest factor will involve four random variables i.e X1, X2, X3, X4.

Complexity of Variable Elimination is determined by the size of the largest factor. So, the order of elimination is important.

If a Bayesian Network is really large and involves 1000s of nodes, even after Variable Elimination, it is still very expensive. So, we adopt Approximate Inference techniques.
.

2 Approximate Inference

In situations where exact inference is computationally expensive, approximate inference methods become valuable.

Sampling is a commonly employed technique. Although there are various sampling techniques the basic algorithm can be summarised as follows:

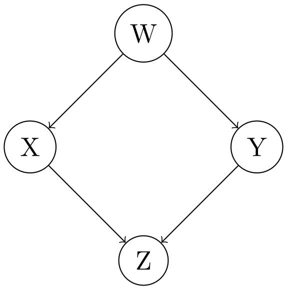
```
For $i = 1$ to $n$ (in topological order)
    Sample $x_i$ from $P(x_i \,|\, \text{parents}(x_i))$
```

Now using the samples of the random variables, the probability can be estimated with their relative counts. Different Sampling methods:

1. Prior Sampling
2. Negative Sampling
3. Likelihood Sampling

2.1 Prior Sampling

Definition: Prior sampling is a technique in probabilistic modeling, specifically Bayesian networks, where samples are generated by iteratively assigning values to variables in topological order based on their conditional distributions given their parents.
Consider the follow example where all the random variables in the Bayesian Network takes binary values as shown below:



The samples drawn in the topological order are:

- z,¬x,¬y,w
- ¬z,x,y,¬w
- ¬z,¬x,y,w
- z,x,y,¬w

QUERY VARIABLE: W

- Total number of samples=4
- Number of samples with w =2
- Number of samples with ¬w =2

So, now P(W)=⟨0.5,0.5⟩
The probability of approximation is closer to the actual probability when more samples are drawn.

2.2 Likelihood Sampling

Definition: Likelihood sampling is a method in Bayesian networks where samples are generated by fixing observed variables (evidence) and sampling the remaining variables based on their likelihood given the fixed evidence.

2.3 Negative Sampling

Definition: Negative sampling is a machine learning strategy commonly employed in training models, particularly for tasks like recommendation systems, where negative instances are sampled to balance training data and improve model efficiency.