Faculty of Computer Science, Dalhousie University *2-Nov-2023*

CSCI 4152/6509 — Natural Language Processing

Lecture 18: Sum-Product (Message-passing) Algorithms for BN Inference

Location: Rowe 1011 Instructor: Vlado Keselj
Time: 16:05 – 17:25

Previous Lecture

- **HMM as Bayesian Network**
- Bayesian Network definition
- Burglar-earthquake example
- Computational tasks
- BN inference using brute force
- Complexity of general inference in BNs
- Sum-product algorithms (started)

Computation Problems Solved by Message Passing

- Applicable to all inference problems
- Two main types of computation:
  - **Summation** of resulting overall products where variables take different domain values
  - **Maximization:** Finding variable values for which the resulting overall product is maximized
- Two main situations:
  - Factor node passing a message to variable node
  - Variable node passing a message to factor node

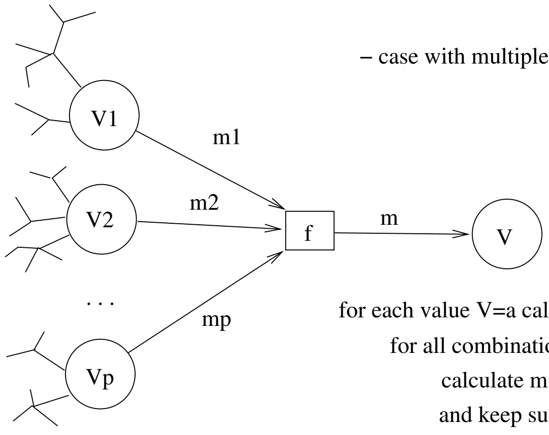
Slide notes:

Four Cases of Message Computation

- Actually, we can distinguish 4 cases of message computation:

1. Factor node with multiple neighbours to variable node
2. Factor leaf node to variable node
3. Variable node with multiple neighbours to factor node
4. Variable leaf node to factor node

Factor Node with Multiple Neighbours Passing a Message to Variable Node

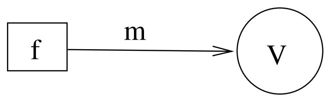


– case with multiple neighbours

```
for each value V=a calculate m(a):
    for all combinations of V1 .. Vp
        calculate m1*m2*..mp*f
        and keep sum or max
    m(a) is resulting sum or max
```

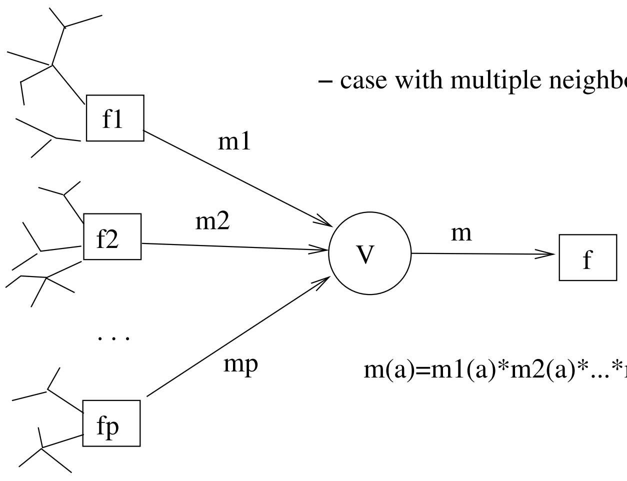
Factor Node with No Other Neighbours Passing a Message to Variable Node

– case with no other neighbours



for each value V=a : m(a) = f(a)

Variable Node with Multiple Neighbours Passing a Message to Factor Node

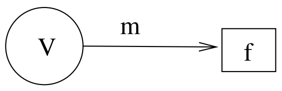


– case with multiple neighbours

m(a)=m1(a)*m2(a)*...*mp(a)

Variable Node with No Other Neighbours Passing a Message to Factor Node

– case with no other neighbours



for each value a of V: m(a) = 1

16.3.1 Solving Inference Tasks with Message-Passing Algorithms

Slide notes:

Solving Inference Tasks

- Distinguish the following cases of inference tasks:

1. Marginalization with one variable
2. Marginalization in general
3. Conditioning with one variable
4. Conditioning in general
5. Completion

Message-Passing Algorithm for Marginalization with One Variable

We first consider the algorithm when we marginalize only on one variable.

Slide notes:

Marginalization with One Variable

- $\mathrm{P}(V_i{=}x_i) = ?$
- Apply general message passing rules with summation
- At the end

$$\mathrm{P}(V_i{=}x_i) \quad = \quad M_{f_1\to V_i}(x_i)\cdots M_{f_p\to V_i}(x_i)$$

- Running time: $O(nm^{p+1})$

- Messages M are vectors of real numbers $(u_1,...,u_m)$, where u_k is a number that summarizes the computation for the case $V = d_k$, where the domain for all variables is $\{d_1, d_2, \ldots, d_m\}$.
- Messages are passed from variable nodes to function nodes, and from function nodes to variable nodes.
- A node can send a message to its neighbor only when it has received all of the messages from its other neighbors.
- Given a tree, the algorithm can start by sending messages from each of the leaves, and stops once every node has passed a message to every neighbor. At the end, two messages will pass each edge in the graph: one for each of the two directions.
- Function to variable messages $M_{f\to V}(x)$ are computed by

$$M_{f\to V}(x) = \sum_{x_1,\ldots,x_p} f(x, x_1, ..., x_p) M_{V_1\to f}(x_1)\cdots M_{V_p\to f}(x_p)$$

over all <u>other</u> variables $V_1, ..., V_p$ (beside V) connected to f. If f is connected only to V, then $M_{f\to V}(x) = f(x)$.

- Variable to function messages $M_{V\to f}(x)$ are computed by

$$M_{V\to f}(x) \quad = \quad \begin{cases} 1 & \text{if only } f \text{ is connected to } V \\ M_{f_1\to V}(x)\cdots M_{f_p\to V}(x) & \text{otherwise} \end{cases}$$

  over all other functions $f_1, ..., f_p$ (beside f) adjacent to V.
- Once all of the messages have been passed, then the final marginal for any variable V_i can be calculated by

$$\mathrm{P}(V_i\!=\!x_i) \quad = \quad M_{f_1\to V_i}(x_i)\cdots M_{f_p\to V_i}(x_i)$$

  for all $f_1, ..., f_p$ adjacent to V_j.

This algorithm is efficient: There are $2n-1$ edges in an undirected tree containing $2n$ nodes (n variables and n function nodes). $2(2n-1)$ messages get sent (one in each direction along each edge). Each function to variable message can be computed in time $O(m^p)$ where p is the number of function neighbors, each variable to function message can be computed in time $O(mp)$ where p is the number of variable neighbors, and the final marginal can be computed in time $O(mp)$. Thus, the total running time is bounded by $O(nm^p)$ where p is the maximum number of neighbors of any node in the graph. This is linear in n and polynomial in m (but exponential in p, so the maximum number of neighbors has to be bounded).

Message-passing Algorithm for Marginalization in General

Slide notes:

Marginalization in General

- Consider calculating $\mathrm{P}(V_1\!=\!x_1, ..., V_k\!=\!x_k)$.
- The variables $V_1, ..., V_k$ are called evidence variables and the instantiated values $x_1, \ldots, x_k$ are called observed evidence.
- An evidence-variable to function message is computed in the same way as before if $x = x_j$ (i.e., it is equal to observed evidence), otherwise it is 0.
- Final computation is done in any evidence node V_j:

$$\mathrm{P}(V_1\!=\!x_1, ..., V_k\!=\!x_k) \quad = \quad M_{f_1\to V_j}(x_j)\cdots M_{f_p\to V_j}(x_j)$$

Consider computing the marginal of one particular partial configuration $\mathrm{P}(V_1\!=\!x_1, ..., V_k\!=\!x_k)$. The variables $V_1, ..., V_k$ are called evidence variables and the instantiated values $x_1, \ldots, x_k$ are called observed evidence. Then we can compute the desired probability by using the same message passing algorithm as above, except:

- An evidence-variable to function message is computed in the same way as before if $x = x_j$ (i.e., it is equal to observed evidence), otherwise it is 0. I.e.,

$$M_{V\to f}(x) \quad = \quad \begin{cases} 0 & \text{if } x \neq x_j \\ 1 & \text{if } x = x_j \text{ and only } f \text{ is adjacent to } V \\ M_{f_1\to V}(x)\cdots M_{f_p\to V}(x) & \text{otherwise } (x = x_j) \end{cases}$$

  over all other functions $f_1, ..., f_p$ (besides f) adjacent to V.
- Once all of the messages have been passed, then the final marginal can be determined by taking any evidence variable $V_j \in \{V_1, ..., V_k\}$ and computing

$$\mathrm{P}(V_1\!=\!x_1, ..., V_k\!=\!x_k) \quad = \quad M_{f_1\to V_j}(x_j)\cdots M_{f_p\to V_j}(x_j)$$

  over all $f_1, ..., f_p$ adjacent to V_j.

Message-passing Algorithm for Conditioning with One Variable

Let us assume that we need to calculate the following conditional probability: $\mathrm{P}(V_{k+1}=y_{k+1}|V_1=x_1,...,V_k=x_k)$. We can use the same message passing algorithm as above, treating $V_1,...,V_k$ as evidence variables, except that

- once all of the messages have been passed, then the final conditional probability can be determined by

$$\begin{aligned}&\mathrm{P}(V_{k+1}=y_{k+1}|V_1=x_1,...,V_k=x_k)\\ =&\ \frac{M_{f_1\to V_{k+1}}(y_{k+1})\cdots M_{f_p\to V_{k+1}}(y_{k+1})}{Z}\end{aligned}$$

  where Z is a normalization constant over choices of V_{k+1}; that is,

$$Z = \sum_y M_{f_1\to V_{k+1}}(y)\cdots M_{f_p\to V_{k+1}}(y)$$

Message-passing Algorithm for Conditioning in General

To compute arbitrary conditional probability $\mathrm{P}(\mathbf{V}_\alpha=\mathbf{y}_\alpha|\mathbf{V}_\beta=\mathbf{x}_\beta)$, where α and β are two disjoint sets of indices from $\{1,\ldots,n\}$, we can use formula:

$$\mathrm{P}(\mathbf{V}_\alpha=\mathbf{y}_\alpha|\mathbf{V}_\beta=\mathbf{x}_\beta)=\frac{\mathrm{P}(\mathbf{V}_\alpha=\mathbf{y}_\alpha,\mathbf{V}_\beta=\mathbf{x}_\beta)}{\mathrm{P}(\mathbf{V}_\beta=\mathbf{x}_\beta)},$$

where we know how to calculate marginal probabilities $\mathrm{P}(\mathbf{V}_\alpha=\mathbf{y}_\alpha,\mathbf{V}_\beta=\mathbf{x}_\beta)$ and $\mathrm{P}(\mathbf{V}_\beta=\mathbf{x}_\beta)$ using the message-passing algorithm.

Message-passing Algorithm for Completion

If we are computing completion with one variable, it is easy to use the algorithm for conditioning on one variable to obtain the result.

However, for completion in general, we apply a new message passing algorithm.

To compute

$$y^*_{k+1},...,y^*_n = \operatorname*{arg\,max}_{y_{k+1},\ldots,y_n} \mathrm{P}(V_{k+1}=y_{k+1},...,V_n=y_n|V_1=x_1,...,V_k=x_k)$$

we can use the same message passing algorithm as the algorithm for calculating marginal probability $\mathrm{P}(V_1=x_1,\ldots,V_k=x_k)$, except:

- Function to variable messages $M_{f\to V}(x)$ are computed by

$$M_{f\to V}(x) = \max_{x_1,\ldots,x_p} f(x,x_1,...,x_p)M_{V_1\to f}(x_1)\cdots M_{V_p\to f}(x_p)$$

  over all other variables $V_1,...,V_p$ (besides V) adjacent to f.
- Once all of the messages have been passed, then the maximum probability completion for any free variable V_{k+j} can be calculated by

$$y^*_{k+j} = \operatorname*{arg\,max}_{y_{k+j}} M_{f_1\to V_{k+j}}(y_{k+j})\cdots M_{f_p\to V_{k+j}}(y_{k+j})$$

  over all $f_1,...,f_p$ containing V_{k+j}.
- If there are two or more values for a variable for which the maximal conditional probability is reached, we need to make sure that all variables are assigned consistently by hard-wiring the chosen variable value.