Faculty of Computer Science, Dalhousie University *30-Oct-2024*

CSCI 4152/6509 — Natural Language Processing

Lecture 15: Inference with HMMs

Location: Carleton Tupper Building Theatre C Instructor: Vlado Keselj
Time: 16:05 – 17:25

Previous Lecture

- Witten-Bell smoothing (finished)
- **POS tagging: Introduction**
- Reading: [JM] Ch5 Part-of-Speech Tagging
- Open word categories
- Closed word categories
- Other word categories
- **Hidden Markov Model (HMM):**
  - idea, definition, graphical representation
  - HMM assumption
- HMM POS Example

Slide notes:

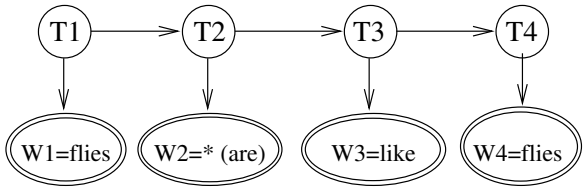
Reminder: Learning HMM (Training)

- Let us Learn HMM from completely labeled data:

```
swat V flies N like P ants N
time N flies V like P an D arrow N
```

- We will use smoothing in word generation, by giving a 0.5 count to all unseen words

Let us use the Hidden Markov Model to POS tag the sentence "flies are like flies."



The problem of POS tagging in this case is the problem of finding the most probable values of the variables T_i given the values of variables W_i, i.e., it is the completion problem of finding

$$\begin{aligned}\arg\max_T \mathrm{P}(T|W=\text{sentence}) &= \arg\max_T \frac{\mathrm{P}(T, W=\text{sentence})}{\mathrm{P}(W=\text{sentence})} = \arg\max_T \mathrm{P}(T, W=\text{sentence}) \\ &= \arg\max_T \mathrm{P}(T_1)\cdot\mathrm{P}(W_1=\text{flies}|T_1)\cdot\mathrm{P}(T_2|T_1)\cdot\mathrm{P}(W_2=*|T_2) \\ &\quad \cdot\mathrm{P}(T_3|T_2)\cdot\mathrm{P}(W_3=\text{like}|T_3)\cdot\mathrm{P}(T_4|T_3)\cdot\mathrm{P}(W_4=\text{flies}|T_4)\end{aligned}$$

where T and W denote arrays of variables T_i and W_i. One way to find the values of T that maximize the given probability is to test all variations of their values. This number is exponential in general, and in this case it is $4^4 = 256$. A much more efficient solution can be obtained by applying a dynamic programming approach, known as the Viterbi algorithm.

"Brute-Force" Approach

- Try all combinations of variable values T_1, T_2, T_3, and T_4
- Calculate the overall probability for each of them using the formula

$$\begin{aligned}&\mathrm{P}(T_1) \cdot \mathrm{P}(W_1 = \text{flies}|T_1)\\ &\cdot\mathrm{P}(T_2|T_1) \cdot \mathrm{P}(W_2 = *|T_2)\\ &\cdot\mathrm{P}(T_3|T_2) \cdot \mathrm{P}(W_3 = \text{like}|T_3)\\ &\cdot\mathrm{P}(T_4|T_3) \cdot \mathrm{P}(W_4 = \text{flies}|T_4)\end{aligned}$$

- Choose the maximal probability

15.3 Efficient Tagging with HMM

Efficient Tagging with HMM

- Rather than using the brute-force approach, we can incrementally optimize the product expression by partial maximization from left to right
- One way to represent this is by using a table, which leads to the dynamic programming solution, or the Viterbi algorithm
- The second way to represent this computation is using message passing, or product-sum algorithm

HMM Inference: Dynamic Programming Solution

- Brute-force approach is too inefficient
- Idea for more efficient calculation: maximize sub-products first
- Dynamic Programming approach: divide problem into sub-problems
  - with a manageable number of sub-problems
- Find maximal partial configurations up to T_1, then T_2, T_3, and T_4

Tagging as Product Maximization

Viterbi Algorithm Example

The idea of the Viterbi algorithm is to incrementally calculate maximal values of the following parts of the above product:

$$\mathrm{P}(T_1) \cdot \mathrm{P}(W_1 = \text{flies}|T_1)$$

for all possible values of T_1, then

$$\mathrm{P}(T_1) \cdot \mathrm{P}(W_1 = \text{flies}|T_1) \cdot \mathrm{P}(T_2|T_1) \cdot \mathrm{P}(W_2 = *|T_2)$$

for all possible values of T_2 and so on. The computation can be summarized in the following table:

	T_1 (W_1 = flies)	T_2 (W_2 = *)	T_3 (W_3 = like)	T_4 (W_4 = flies)
	$\mathrm{P}(T_1)\mathrm{P}(W_1\|T_1)$	$p \cdot \mathrm{P}(T_2\|T_1)\mathrm{P}(W_2\|T_2)$	$p \cdot \mathrm{P}(T_3\|T_2)\mathrm{P}(W_3\|T_3)$	$p \cdot \mathrm{P}(T_4\|T_3)\mathrm{P}(W_4\|T_4)$
D	$0 \times 0 = 0$	DD: $0 \times 0 \times \frac{1}{3} = 0$ ND: $\frac{1}{9} \times 0 \times \frac{1}{3} = 0$ PD: 0 VD: 0 max: 0	DD: $0 \times 0 \times 0 = 0$ ND: $\frac{1}{90} \times 0\times = 0$ PD: $\frac{1}{50} \times \frac{1}{2} \times 0 = 0$ VD: $\frac{1}{90} \times 0 \times 0 = 0$ max: 0	DD: $0 \times 0 \times 0 = 0$ ND: $0 \times 0 \times 0 = 0$ PD: $\frac{1}{225} \times 0.5 \times 0 = 0$ VD: $0 \times 0 \times 0 = 0$ max: 0
N	$0.5 \times \frac{2}{9} = \frac{1}{9}$	DN: $0 \times 1 \ldots = 0$ NN: $\frac{1}{9} \times 0 \ldots = 0$ PN: $0 \times \ldots = 0$ VN: $0.2\times0.5\times\frac{1}{9} = \frac{1}{90}$ max: $\frac{1}{90}$	DN: $0 \times 1 \times 0 = 0$ NN: $\frac{1}{90} \times 0 \ldots = 0$ PN: $\frac{1}{50}\times0.5\times0 = 0$ VN: $\frac{1}{90} \times 0.5 \times 0 = 0$ max: 0	DN: $0 \times 1 \times \frac{2}{9} = 0$ NN: $0 \times 0 \times \frac{2}{9} = 0$ PN: $\frac{1}{225}\times0.5\times\frac{2}{9} = \frac{1}{2025}$ VN: $0 \times 0.5 \times \frac{2}{9} = 0$ max: $\frac{1}{2025}$
P	$0 \times 0 = 0$	DP: $0 \times \ldots = 0$ NP: $\frac{1}{9}\times0.5\times0.2 = \frac{1}{90}$ PP: $0 \times \ldots = 0$ VP: $0.2\times0.5\times0.2 = \frac{1}{50}$ max: $\frac{1}{50}$	DP: $0 \times 0 \times 0.8 = 0$ NP: $\frac{1}{90} \times 0.5 \times 0.8 = \frac{1}{225}$ PP: $\frac{1}{50} \times 0 \times 0.8 = 0$ VP: $\frac{1}{90}\times0.5\times0.8 = \frac{1}{225}$ max: $\frac{1}{225}$	DP: $0 \times 0 \times 0 = 0$ NP: $0 \times 0.5 \times 0 = 0$ PP: $\frac{1}{225} \times 0 \times 0 = 0$ VP: $0 \times 0.5 \times 0 = 0$ max: 0
V	$0.5 \times 0.4 = 0.2$	DV: $0 \times \ldots = 0$ NV: $\frac{1}{9}\times0.5\times0.2 = \frac{1}{90}$ PV: $0 \times \ldots = 0$ VV: $0.2 \times 0 \ldots = 0$ max: $\frac{1}{90}$	DV: $0 \times 0 \times 0 = 0$ NV: $\frac{1}{90} \times 0.5 \times 0 = 0$ PV: $\frac{1}{50} \times 0 \times 0 = 0$ VV: $\frac{1}{90} \times 0 \times 0 = 0$ max: 0	DV: $0 \times 0 \times 0.4 = 0$ NV: $0 \times 0.5 \times 0.4 = 0$ PV: $\frac{1}{225} \times 0 \times 0.4 = 0$ VV: $0 \times 0 \times 0.4 = 0$ max: 0

The table is filled column by column. We can see now that the largest value that the expression

$$\mathrm{P}(T_1) \cdot \mathrm{P}(W_1 = \text{flies}|T_1) \cdot \cdot \mathrm{P}(T_2|T_1) \cdot \mathrm{P}(W_2 = *|T_2) \cdot \mathrm{P}(T_3|T_2) \cdot \mathrm{P}(W_3 = \text{like}|T_3) \cdot \mathrm{P}(T_4|T_3) \cdot \mathrm{P}(W_4 = \text{flies}|T_4)$$

can obtain is $\frac{1}{2025}$, and is achieved with $T_4 = \mathrm{N}$. If we work backwards through the table, we can obtain the optimal values for previous variables as well: $T_3 = \mathrm{P}$, $T_2 = \mathrm{V}$, and $T_1 = \mathrm{N}$. We can also choose $T_2 = \mathrm{N}$, but in this case we have $T_1 = \mathrm{V}$.

16 HMM as Bayesian Network

Slide notes:

HMM as Bayesian Network

- **Viterbi** algorithm is an **efficient** way to solve a **special** problem:
  - completion with known observables and unknown hidden nodes of an HMM
- **General** approach:
  - Treat HMM as **Bayesian Network**
  - Apply **Product-Sum** (i.e., "Message-passing") algorithm for efficient inference

In the previous section, we saw how we can do efficient inference in an HMM model using the Viterbi algorithm. The inference task that we were solving was a specialized completion task where we knew values of all observable variables, and we calculated most likely values of all hidden variables. In order to solve more general inference tasks, such as marginalization, or for example completion with arbitrary known variables, we would need to modify the Viterbi algorithm in a non-trivial way, or possibly come up with a significantly different new algorithm.

A much more general and flexible way to approach inference in HMMs is to treat them as Bayesian Networks, since HMM is a specialized version of a more general model called *Bayesian Network*.

16.1 Bayesian Networks

Note regarding course scope: Some detailed discussion in this section is not completely covered in the lectures, so that part will not be used in assignments or the exam. For the purposes of the course, it is sufficient to understand how product-sum algorithms work and be able to apply them or implement them.

Slide notes:

Bayesian Network Model

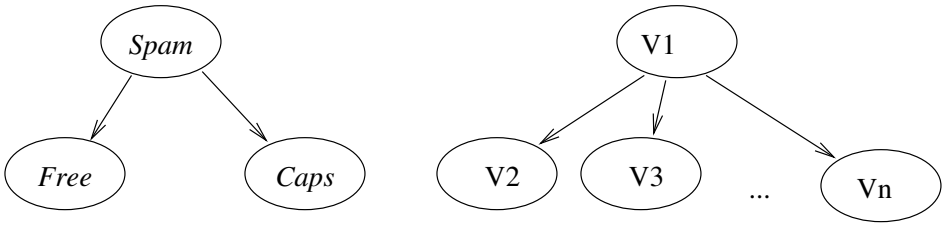
- Also known as: Belief Networks, or Bayesian Belief Networks
- A directed acyclic graph (DAG)
  - Each node representing a random variable
  - Edges representing causality (probabilistic meaning)
- Conditional Probability Table (CPT) for each node
- Bayesian Network assumption:

$$\mathrm{P}(\text{ full configuration }) = \prod_{i=1}^{n} \mathrm{P}(V_i|\mathbf{V}_{\pi(i)})$$

The Viterbi algorithm provides an efficient way to solve a particular inference problem in the HMM model. It is the completion problem of finding the most likely hidden states given the observable values. If we represent the HMM model in the unrolled graphical form, we can see that this inference problem is only a special case of the inference problem, and that we may want to calculate other inference formulas, including different marginalization, conditioning, or completion cases. Many of these problems can be efficiently solved if we see HMM as a special case of a more general concept of a Bayesian Network.

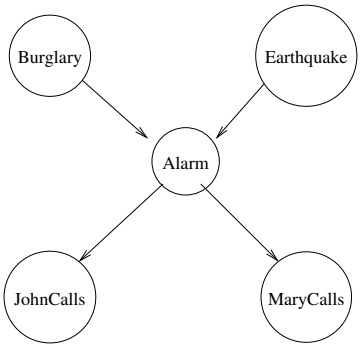
Bayesian Networks (also known as **belief networks, Bayesian belief networks,** or **decision networks**) provide a way to create structured probabilistic models in a more general way. We saw previously that between the joint distribution model and the fully independent model we have useful models such as the Naïve Bayes model, the Ngram or Markov chain model, and the Hidden Markov Model. All of these models had a graphical representation, and the Bayesian Network model makes this more general by providing a model associated with any directed acyclic graph.

The following graphs shows dependence structure for the Naïve Bayes model of the spam detection, and for the Naïve Bayes model in general:



Example

To introduce the general concept of Bayesian Networks, we will consider the known Burglar-Earthquake example, frequently used in the literature (e.g., the AI textbook by Russell and Norvig):



We can follow a topological sort of the above graph and use the chain rule for the conditional probability, which can be derived from the definition of the conditional probability, to obtain the equation:

$$\mathrm{P}(B, E, A, J, M) = \mathrm{P}(B)\mathrm{P}(E|B)\mathrm{P}(A|B, E)\mathrm{P}(J|A, B, E)\mathrm{P}(M|J, A, B, E)$$

The graph denotes some dependence assumptions, which include: B and E are independent variables, hence $\mathrm{P}(E|B) = \mathrm{P}(E)$, J depends only on A, hence $\mathrm{P}(J|A, B, E) = \mathrm{P}(J|A)$, and M depends only on A, hence $\mathrm{P}(M|J, A, B, E) = \mathrm{P}(M|A)$. If we make these substitutions in the above equation, we obtain the following, **Bayesian network assumption:**

$$\mathrm{P}(B, E, A, J, M) = \mathrm{P}(B)\mathrm{P}(E)\mathrm{P}(A|B, E)\mathrm{P}(J|A)\mathrm{P}(M|A)$$

This assumption implies that to evaluate any probability of a complete configuration of the model we need to keep only the parameters for the following conditional probabilities: $\mathrm{P}(B)$, $\mathrm{P}(E)$, $\mathrm{P}(A|B, E)$, $\mathrm{P}(J|A)$, and $\mathrm{P}(M|A)$. These parameters are given in corresponding **conditional probability tables** (CPTs).

Let us assume that the conditional tables for our example are:

B	$\mathrm{P}(B)$
T	0.001
F	0.999

E	$\mathrm{P}(E)$
T	0.002
F	0.998

B	E	A	$\mathrm{P}(A\|B, E)$
T	T	T	0.95
T	T	F	0.05
T	F	T	0.94
T	F	F	0.06
F	T	T	0.29
F	T	F	0.71
F	F	T	0.001
F	F	F	0.999

A	J	$\mathrm{P}(J\|A)$
T	T	0.90
T	F	0.10
F	T	0.05
F	F	0.95

A	M	$\mathrm{P}(M\|A)$
T	T	0.70
T	F	0.30
F	T	0.01
F	F	0.99

The following is a formal definition of Bayesian Network:

Definition 16.1 (Bayesian Network) *A Bayesian Network is defined by a directed acyclic graph (DAG) and a collection of conditional probability tables, where nodes in the graph represent random variables and directed edges in the graph represent conditional independence assumptions. The edges represent causality among the random variable nodes so that there is an edge from a variable to another variable if and only if there is a direct dependence between these variables. If V_j ($1 \leq j \leq n$) are the random variables and $\mathbf{V}_{\pi(j)}$ are parent variables of any variable V_j, i.e., then the Bayesian Network assumption can be written in the following way:*

$$\mathrm{P}(\textit{ all variables }) = \prod_{j=1}^{n} \mathrm{P}(V_j|\mathbf{V}_{\pi(j)})$$

Hence, a Bayesian network consists of two components: a directed graph and a set of lookup tables for conditional probabilities for each variable V_i. If $\pi(i)$ are parent nodes of V_i, then for all possible values of variables $\mathbf{V}_{\pi(i)}$ and V_i, the conditional probability table (CPT) specifies the probability of $\mathrm{P}(V_i|\mathbf{V}_{\pi(i)})$.

Example (Burglar-Earthquake cont'd)

The tables for the Burglar-Earthquake example:

B	$\mathrm{P}(B)$
T	0.001
F	0.999

E	$\mathrm{P}(E)$
T	0.002
F	0.998

B	E	A	$\mathrm{P}(A\|B,E)$
T	T	T	0.95
T	T	F	0.05
T	F	T	0.94
T	F	F	0.06
F	T	T	0.29
F	T	F	0.71
F	F	T	0.001
F	F	F	0.999

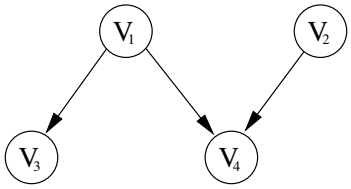
A	J	$\mathrm{P}(J\|A)$
T	T	0.90
T	F	0.10
F	T	0.05
F	F	0.95

A	M	$\mathrm{P}(M\|A)$
T	T	0.70
T	F	0.30
F	T	0.01
F	F	0.99

Number of Parameters

If the number of parent nodes is k, and if we assume that each variable may have m different values, than the conditional probability table has m^{k+1} rows. For each of m^k combinations of values of parent notes, there is one constraint on the probability distribution $\sum_x \mathrm{P}(V_i = x|\mathbf{V}_{\pi(i)}) = 1$, which will result in m^k constraints; i.e., there are $m^{k+1} - m^k = (m-1)m^k$ free parameters. If the maximal number of parents for a whole network is k, then the total number of free parameters is not grater than $n(m-1)m^k$.

Example. For example, let us calculate the number of free parameters of the following Bayesian Network:



The Bayesian assumption for the network above is:

$$\begin{aligned}&\mathrm{P}(V_1{=}x_1, V_2{=}x_2, V_3{=}x_3, V_4{=}x_4)\\ &\quad = \quad \mathrm{P}(V_1{=}x_1)\ \mathrm{P}(V_2{=}x_2)\ \mathrm{P}(V_3{=}x_3|V_1{=}x_1)\ \mathrm{P}(V_4{=}x_4|V_1{=}x_1, V_2{=}x_2)\end{aligned}$$

How many parameters are needed to represent the network?

For each variable store a conditional probability table of size

$$m \cdot m^{\#\text{parents}} - m^{\#\text{parents}} \quad \text{(constraints)}$$

For the above network, the number of free parameters is:

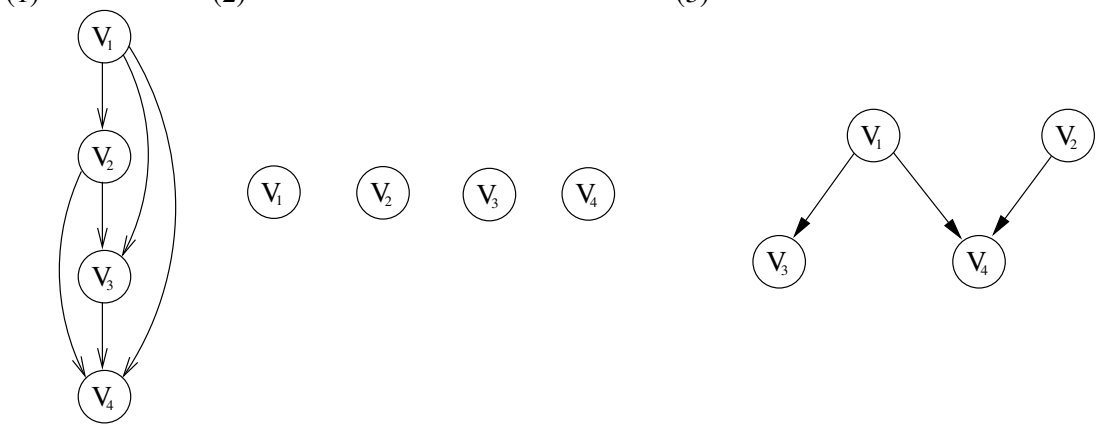
$$\begin{aligned}&m + m + m^2 + m^3 \quad \text{parameters}\\ &-1 - 1 - m - m^2 \quad \text{constraints}\\ &= m^3 + m - 2\end{aligned}$$

Representational Power

Using the Bayesian network model we can represent many other models: the full joint distribution model, fully independent model, Naïve Bayes model, Hidden Markov Model, and other structured models.

Some examples of the Bayesian Networks are:

(1) (2) (3)



Number of parameters in each model:

(1) $(m-1)+(m^2-m)+(m^3-m^2)+(m^4-m^3)=m^4-1$

(2) $(m-1)+(m-1)+(m-1)+(m-1)=4m-4$

(3) solved above: m^3+m-2

16.2 Computational Tasks

Evaluation

The calculate probability of a complete configuration, we multiply corresponding conditional probabilities:

$$\mathrm{P}(V_1{=}x_1,...,V_n{=}x_n) \quad = \quad \prod_{i=1}^{n} \mathrm{P}(V_i{=}x_i|\mathbf{V}_{\pi(i)}{=}\mathbf{x}_{\pi(i)})$$

Simulation

For $i = 1,...,n$, draw x_j according to $\mathrm{P}(V_j = x_j|\mathbf{V}_{\pi(j)} = \mathbf{x}_{\pi(j)})$. Conjoin $(x_1,...,x_n)$ to form a complete configuration.

Learning

Learning with predetermined network graph, and from complete observations can be done by direct MLE; i.e., counting.

Inference in Bayesian Networks

One way to handle all inference tasks in Bayesian Networks is to use the brute force method. Using the same summation techniques as in the Joint distribution method, we can do marginalization, conditioning, and completion in Bayesian Networks as well. Although this method does not necessarily use a lot of memory, it is prohibitively expensive in terms of running time. The running time is the same as in the Joint distribution model, i.e., it is exponential. For example, there are $2^5 = 512$ possible configurations in the Burglar-Earthquake example network.

Inference Example using Brute Force

By using the tables we can easily compute the probability of any complete configuration. With appropriate summations and by using the definitions of marginal and conditional probability, we can also solve other inference problems. For example, if we want to calculate $\mathrm{P}(B = T|J = T)$, i.e., the probability that a burglar is in the house if John told us over the phone that the alarm is on, we first use the definition of the conditional probability:

$$\mathrm{P}(B = T|J = T) = \frac{\mathrm{P}(B = T, J = T)}{\mathrm{P}(J = T)}$$

The marginal probability $\mathrm{P}(B = T, J = T)$ can be calculated using the formula:

$$\begin{aligned}\mathrm{P}(B = T, J = T) &= \sum_{E,A,M} \mathrm{P}(B = T, E, A, J = T, M) \\ &= \sum_{E,A,M} \mathrm{P}(B = T)\mathrm{P}(E)\mathrm{P}(A|B = T, E)\mathrm{P}(J = T|A)\mathrm{P}(M|A)\end{aligned}$$

Hence,

$$\begin{aligned}\mathrm{P}(B = T, J = T) &= \\ &\quad \mathrm{P}(B = T)\mathrm{P}(E = T)\mathrm{P}(A = T|B = T, E = T)\mathrm{P}(J = T|A = T)\mathrm{P}(M = T|A = T) \\ &+ \mathrm{P}(B = T)\mathrm{P}(E = T)\mathrm{P}(A = T|B = T, E = T)\mathrm{P}(J = T|A = T)\mathrm{P}(M = F|A = T) \\ &+ \mathrm{P}(B = T)\mathrm{P}(E = T)\mathrm{P}(A = F|B = T, E = T)\mathrm{P}(J = T|A = F)\mathrm{P}(M = T|A = F) \\ &+ \mathrm{P}(B = T)\mathrm{P}(E = T)\mathrm{P}(A = F|B = T, E = T)\mathrm{P}(J = T|A = F)\mathrm{P}(M = F|A = F) \\ &+ \mathrm{P}(B = T)\mathrm{P}(E = F)\mathrm{P}(A = T|B = T, E = F)\mathrm{P}(J = T|A = T)\mathrm{P}(M = T|A = T) \\ &+ \mathrm{P}(B = T)\mathrm{P}(E = F)\mathrm{P}(A = T|B = T, E = F)\mathrm{P}(J = T|A = T)\mathrm{P}(M = F|A = T) \\ &+ \mathrm{P}(B = T)\mathrm{P}(E = F)\mathrm{P}(A = F|B = T, E = F)\mathrm{P}(J = T|A = F)\mathrm{P}(M = T|A = F) \\ &+ \mathrm{P}(B = T)\mathrm{P}(E = F)\mathrm{P}(A = F|B = T, E = F)\mathrm{P}(J = T|A = F)\mathrm{P}(M = F|A = F) \\ &= 0.001 \cdot 0.002 \cdot 0.95 \cdot 0.9 \cdot 0.7 \\ &+ 0.001 \cdot 0.002 \cdot 0.95 \cdot 0.9 \cdot 0.3 \\ &+ 0.001 \cdot 0.002 \cdot 0.05 \cdot 0.05 \cdot 0.01 \\ &+ 0.001 \cdot 0.002 \cdot 0.05 \cdot 0.05 \cdot 0.99 \\ &+ 0.001 \cdot 0.998 \cdot 0.94 \cdot 0.9 \cdot 0.7 \\ &+ 0.001 \cdot 0.998 \cdot 0.94 \cdot 0.9 \cdot 0.3 \\ &+ 0.001 \cdot 0.998 \cdot 0.06 \cdot 0.05 \cdot 0.01 \\ &+ 0.001 \cdot 0.998 \cdot 0.06 \cdot 0.05 \cdot 0.99 \\ &= 8.49017 \cdot 10^{-4}\end{aligned}$$

To calculate $\mathrm{P}(J = T)$, we can represent it as $\mathrm{P}(J = T) = \mathrm{P}(B = T, J = T) + \mathrm{P}(B = F, J = T)$ and first

calculate $\mathrm{P}(B = F, J = T)$:

$$\begin{aligned}
\mathrm{P}(B = F, J = T) &= \sum_{E,A,M} \mathrm{P}(B = F, E, A, J = T, M) \\
&= \sum_{E,A,M} \mathrm{P}(B = F)\mathrm{P}(E)\mathrm{P}(A|B = F, E)\mathrm{P}(J = T|A)\mathrm{P}(M|A) \\
&= \mathrm{P}(B = F)\mathrm{P}(E = T)\mathrm{P}(A = T|B = F, E = T)\mathrm{P}(J = T|A = T)\mathrm{P}(M = T|A = T) \\
&+ \mathrm{P}(B = F)\mathrm{P}(E = T)\mathrm{P}(A = T|B = F, E = T)\mathrm{P}(J = T|A = T)\mathrm{P}(M = F|A = T) \\
&+ \mathrm{P}(B = F)\mathrm{P}(E = T)\mathrm{P}(A = F|B = F, E = T)\mathrm{P}(J = T|A = F)\mathrm{P}(M = T|A = F) \\
&+ \mathrm{P}(B = F)\mathrm{P}(E = T)\mathrm{P}(A = F|B = F, E = T)\mathrm{P}(J = T|A = F)\mathrm{P}(M = F|A = F) \\
&+ \mathrm{P}(B = F)\mathrm{P}(E = F)\mathrm{P}(A = T|B = F, E = F)\mathrm{P}(J = T|A = T)\mathrm{P}(M = T|A = T) \\
&+ \mathrm{P}(B = F)\mathrm{P}(E = F)\mathrm{P}(A = T|B = F, E = F)\mathrm{P}(J = T|A = T)\mathrm{P}(M = F|A = T) \\
&+ \mathrm{P}(B = F)\mathrm{P}(E = F)\mathrm{P}(A = F|B = F, E = F)\mathrm{P}(J = T|A = F)\mathrm{P}(M = T|A = F) \\
&+ \mathrm{P}(B = F)\mathrm{P}(E = F)\mathrm{P}(A = F|B = F, E = F)\mathrm{P}(J = T|A = F)\mathrm{P}(M = F|A = F) \\
&= 0.999 \cdot 0.002 \cdot 0.29 \cdot 0.9 \cdot 0.7 \\
&+ 0.999 \cdot 0.002 \cdot 0.29 \cdot 0.9 \cdot 0.3 \\
&+ 0.999 \cdot 0.002 \cdot 0.71 \cdot 0.05 \cdot 0.01 \\
&+ 0.999 \cdot 0.002 \cdot 0.71 \cdot 0.05 \cdot 0.99 \\
&+ 0.999 \cdot 0.998 \cdot 0.001 \cdot 0.9 \cdot 0.7 \\
&+ 0.999 \cdot 0.998 \cdot 0.001 \cdot 0.9 \cdot 0.3 \\
&+ 0.999 \cdot 0.998 \cdot 0.999 \cdot 0.05 \cdot 0.01 \\
&+ 0.999 \cdot 0.998 \cdot 0.999 \cdot 0.05 \cdot 0.99 \\
&= 5.12899587 \cdot 10^{-2}
\end{aligned}$$

Now, we calculate

$$\mathrm{P}(J = T) = \mathrm{P}(B = T, J = T) + \mathrm{P}(B = F, J = T) = 8.49017 \cdot 10^{-4} + 5.12899587 \cdot 10^{-2} = 0.0521389757,$$

and finally

$$\mathrm{P}(B = T|J = T) = \frac{\mathrm{P}(B = T, J = T)}{\mathrm{P}(J = T)} = \frac{8.49017 \cdot 10^{-4}}{0.0521389757} = 0.0162837299467699.$$

Even this small example illustrates inefficiency of this approach.

Complexity of General Inference in Bayesian Networks

In general, we know that for some Bayesian networks the inference must be costly: for example, inference with an arbitrary joint distribution model is expensive, and a joint distribution model can be regarded as a special case of a Bayesian network. The size of the joint distribution model is exponentially large due to the large m^n look up table (m is the size of the domain, and n is the number of random variables).

On the other hand, inference in some very simple Bayesian networks such as the fully independent model and the Naïve Bayes model can be done efficiently. Inference in HMMs can also be done efficiently. An interesting question is: Can we find some other networks where inference is also easy?

We could first notice that the number of parents per node is limited in the special Bayesian networks that we mentioned: in the fully independent model all notes have 0 parents, in the Naïve Bayes model all nodes have at

most 1 parent, and in the HMM each node has at most 1 parent as well. So, the first idea is to restrict the maximal number of parents in each node.

If the maximal number of parents is 1, then inference can be done efficiently, and we will see this soon. If the maximal number of parents is 2, inference can be hard. The representation size (the table size) is still polynomial, but the inference task can be NP-hard. We can prove that inference in Bayesian Networks with at most 2 parents is NP-hard by the NP-reduction from the Circuit satisfiability problem, or, equivalently, to satisfiability of an arbitrary boolean function. A sketch of this proof is as follows: If we have an arbitrary circuit satisfiability problem it can be represented as a directional circuit with some 0/1 inputs, boolean gates (AND, OR, and NOT), and an output. The satisfiability problem is the question of whether it is possible to set input nodes to the values 0 or 1 in such a way that we obtain 1 at the output node. We can turn this circuit into a Bayesian Network but attaching a node to each input and assigning probability 0.5 to 0 or 1 values. The probability tables are simple 0-1 tables corresponding to associated boolean operations. Satisfiability of the circuit is then equivalent to verifying that the marginal probability of the output node is greater than zero.

We will show that a Bayesian network in which each node has at most one parent node allows for efficient inference. Actually, more generally, if the Bayesian network graph has a shape of the tree; i.e., its undirected version of the graph is a tree, then we know how to do inference efficiently.

16.3 Sum-Product Algorithms for Bayesian Networks

Slide notes:

Sum-Product Algorithms for Bayesian Networks

- Basic idea: optimizing sum-product calculation using graph structure
  Described in "Factor graphs and the Sum-Product Algorithm" by Kschishang, Frey, and Loeliger in 2000
- Algorithm overview:
  1. Construction of a factor graph
  2. Message-passing algorithms
- Construction of the factor graph
- Principles of message passing

The algorithms for inference (marginalization, conditioning, and completion) in tree Bayesian networks can be formulated conveniently as message passing algorithms, also known as product-sum algorithms. The algorithm was re-invented in different disguises in Hidden Markov Models, Kalman filters, Error-correcting codes, Bayesian Networks, and similar.[2]

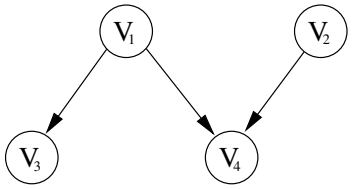
Before presenting the message passing algorithms, let us first see on an example how we can do marginalization efficiently on a Bayesian network, using the BN assumption and the sum separation rule:

Example

Note: This example is not covered in class.

This example illustrates the underlying mathematical computation that leads to the message passing algorithms that we will cover. Let us assume that we have the following Bayesian Network:

[2] The paper introducing the message passing algorithms is "Factor Graphs and the Sum-Product Algorithm" by Frank R. Kschishang, Brendan J. Frey, and Hans-Andrea Loeliger in 2000.



and that we want to compute $P(V_4 = x_4)$:

$$\begin{aligned} &= \sum_{x_1}\sum_{x_2}\sum_{x_3} P(V_1 = x_1, V_2 = x_2, V_3 = x_3, V_4 = x_4) \\ &= \sum_{x_1}\sum_{x_2}\sum_{x_3} P(V_1 = x_1)P(V_2 = x_2)P(V_3 = x_3|V_1 = x_1) \\ &\qquad P(V_4 = x_4|V_1 = x_1, V_2 = x_2) \end{aligned}$$

This computation has $m^3 - 1$ additions and $3m^3$ multiplications, which is very expensive (as expensive as the joint distribution model).

Let us think of conditional probabilities as functions:

$$\begin{aligned} f_1(x_1) &= P(V_1 = x_1) \\ f_2(x_2) &= P(V_2 = x_2) \\ f_3(x_1, x_3) &= P(V_3 = x_3|V_1 = x_1) \\ f_4(x_1, x_2, x_4) &= P(V_4 = x_4|V_1 = x_1, V_2 = x_2) \end{aligned}$$

and let us try to calculate the value of the above expression with less additions and multiplications. We use the fact that summations of factors which depend on variables x_2 and x_3 can be separated.

$$\begin{aligned} &P(V_4 = x_4) = \\ &= \sum_{x_1}\sum_{x_2}\sum_{x_3} f_1(x_1)f_2(x_2)f_3(x_1, x_3)f_4(x_1, x_2, x_4) \\ &= \sum_{x_1} f_1(x_1) \sum_{x_3} f_3(x_1, x_3) \sum_{x_2} f_2(x_2)f_4(x_1, x_2, x_4) \\ &= \sum_{x_1} f_1(x_1) \left(\sum_{x_2} f_2(x_2)f_4(x_1, x_2, x_4) \right) \sum_{x_3} f_3(x_1, x_3) \end{aligned}$$

We can calculate number of operations: First, for each value of x_1 we make m multiplications in sum over x_2, $m - 1$ additions in sum over x_2, $m - 1$ additions in sum over x_3, and two more multiplications of these two sums and f_1, which adds up to $m + 2$ multiplications and $2m - 2$ additions. This is repeated m times, which makes $m^2 + 2m$ multiplications and $2m^2 - 2m$ additions. After that, we make additional $m - 1$ additions, which makes $m^2 + 2m$ multiplications, and $2m^2 - m - 1$ additions.

This is significantly better than $3m^3$ multiplications and $m^3 - 1$ additions.

Breaking up a brute-force formula as in this example into more efficient computations gets more complicated and harder to manage in general Bayesian Networks. It can be done relatively efficiently in tree-structured Bayesian Networks; i.e., networks that do not have cycles when they graphs are treated as undirected graphs.

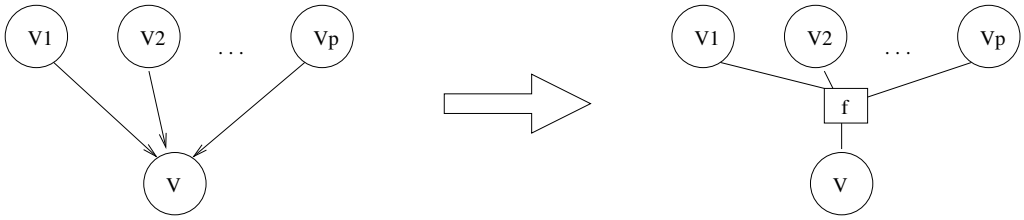
Message Passing Algorithms Framework

Applying the message passing algorithms to Bayesian Network inference consists of two steps:

1. translation of a BN into a factor graph, and
2. message exchange between nodes in the factor graph.

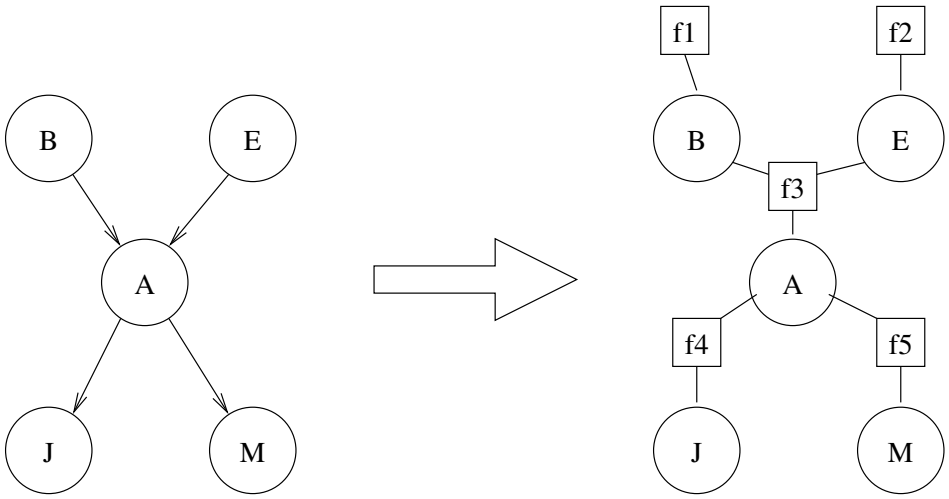
Factor graph

The first step in Bayesian network inference is to convert the Bayesian network graph into a **factor graph**. A factor graph is an an undirected graph in which each variable is represented with one "round" node and one "square" node representing the corresponding conditional probability table (CPT). The square node is connected with all variables appearing in CPT. So, if V is a variable in a Bayesian network, the following transformation is applied.



Example:

Using the Burglar-Earthquake example:

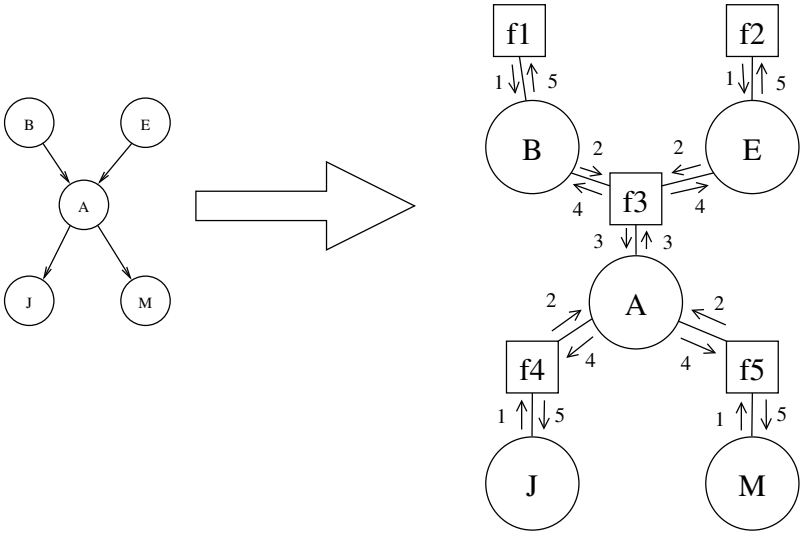


Principles of Message Passing

- A message summarizes computation in the corresponding part of graph
- Messages are vectors of real numbers
- Each node passes to each neighbour node a message exactly once
- To pass a message to a neighbour node, a node needs to receive messages from all other neighbour nodes
- Important property: a tree-structured Bayesian Network leads to a tree factor graph

An important property of this transformation is that if the original Bayesian Network is a tree then the resulting factor graph will also be a tree.

Message Passing Example: Order of Message Computation



The basic overall concept captured with a factor graph with nodes $V_1, ..., V_n$ and factors (or functions) $f_1, ..., f_k$, is that the probability (or some other form of calculation result) of the whole configuration is calculated as the product of factors, which depend on variables, which they are connected to; i.e.,

$$\mathrm{P}(V_1 = x_1, ..., V_n = x_n) \quad = \quad \prod_{j=1}^{k} f_j(\mathbf{x}_j)$$

where $\mathbf{x}_j =$ the values of variables connected to f_j.

After constructing the factor graph, we can handle each inference task using a message passing algorithm.

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