Natural Language Processing CSCI 4152/6509 — Lecture 15 Inference with HMMs

Instructors: Vlado Keselj
Time and date: 16:05 – 17:25, 30-Oct-2024
Location: Carleton Tupper Building Theatre C

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

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

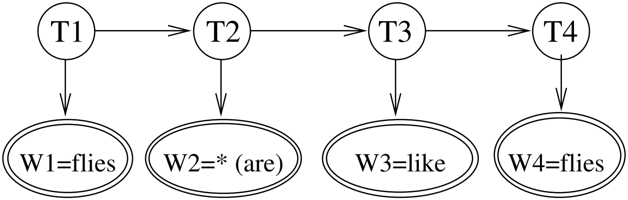
Reminder: Generated Tables

T_1	$P(T_1)$
N	0.5
V	0.5

T_{i-1}	T_i	$P(T_i \mid T_{i-1})$
D	N	1
N	P	0.5
N	V	0.5
P	D	0.5
P	N	0.5
V	N	0.5
V	P	0.5

T_i	W_i	$P(W_i \mid T_i)$
D	an	$2/3 \approx 0.666666667$
D	*	$1/3 \approx 0.333333333$
N	ants	$2/9 \approx 0.222222222$
N	arrow	$2/9 \approx 0.222222222$
N	flies	$2/9 \approx 0.222222222$
N	time	$2/9 \approx 0.222222222$
N	*	$1/9 \approx 0.111111111$
P	like	0.8
P	*	0.2
V	flies	0.4
V	swat	0.4
V	*	0.2

Tagging Example



$$\arg\max_T \mathrm{P}(T|W = \textsf{sentence}) =$$

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

"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

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

- Choose the maximal probability

Brute-Force Approach (tabular view)

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

Dynamic Programming Approach (graphical view)

Viterbi Algorithm Example

	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: $0.2\times 0.5\times\frac{1}{9}=\frac{1}{90}$ max: $\frac{1}{90}$	DN: $0\times 1\times 0=0$ NN: $\frac{1}{90}\times 0\ldots=0$ PN: $\frac{1}{50}\times 0.5\times 0=0$ 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: $\frac{1}{225}\times 0.5\times\frac{2}{9}=\frac{1}{2025}$ 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: $\frac{1}{9}\times 0.5\times 0.2=\frac{1}{90}$ PP: $0\times\ldots=0$ VP: $0.2\times 0.5\times 0.2=\frac{1}{50}$ 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: $\frac{1}{90}\times 0.5\times 0.8=\frac{1}{225}$ 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: $\frac{1}{9}\times 0.5\times 0.2=\frac{1}{90}$ 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

HMM as Bayesian Network

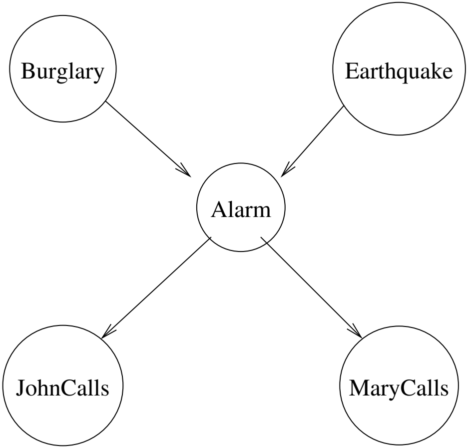
- **Viterbi** algorithm is an **efficient** way to solve a **special** problem:
  - completion with known observables and unknown hidden nodes of an HMM
- **General** approach:
  - Treat HMM as **Bayesian Network**
  - Apply **Product-Sum** (i.e., "Message-passing") algorithm for efficient inference

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:

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

Bayesian Network Example



Bayesian Network Assumption

- Bayesian Network Assumption for previous example:

$$P(B, E, A, J, M) = P(B)P(E)P(A|B, E)P(J|A)P(M|A)$$

- Probability of a complete configuration is a product of conditional probabilities
- Each node corresponds to one conditional probability: $P(B)$, $P(E)$, $P(A|B, E)$, $P(J|A)$, $P(M|A)$
- CPTs (Conditional Probability Tables are model parameters)

Conditional Probability Tables

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

Computational Tasks

- Evaluation:

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

- Simulation
- Learning from complete observations
- Inference in Bayesian Networks

Inference Example using Brute Force

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

$$\begin{aligned} P(B = T, J = T) &= \sum_{E,A,M} P(B = T, E, A, J = T, M) \\ &= \sum_{E,A,M} P(B = T)P(E)P(A|B = T, E) \\ &\quad P(J = T|A)P(M|A) \\ &\approx 8.49017 \cdot 10^{-4} \end{aligned}$$

(continued)

$$P(J = T) = P(B = T, J = T) + P(B = F, J = T)$$

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

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

General Inference in Bayesian Networks

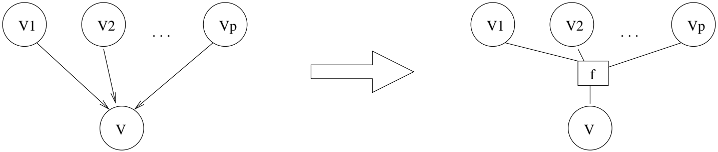
- In some Bayesian Networks inference is always expensive; e.g., joint distribution has a very large number of parameters
- Can we be more efficient if number of parent nodes is limited?
- Naïve Bayes or HMM has a limit of parents to 1
- If we limit number of parents to 2, this may already lead to an NP-hard inference problem
- Proof: a reduction from Circuit Satisfiability problem

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

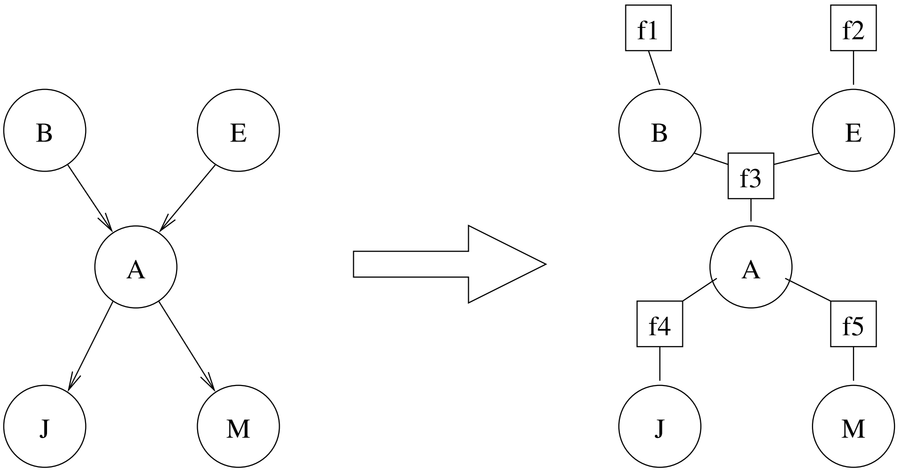
Factor Graph

- Introduce factor nodes:



- Factor graph captures the structure of computation

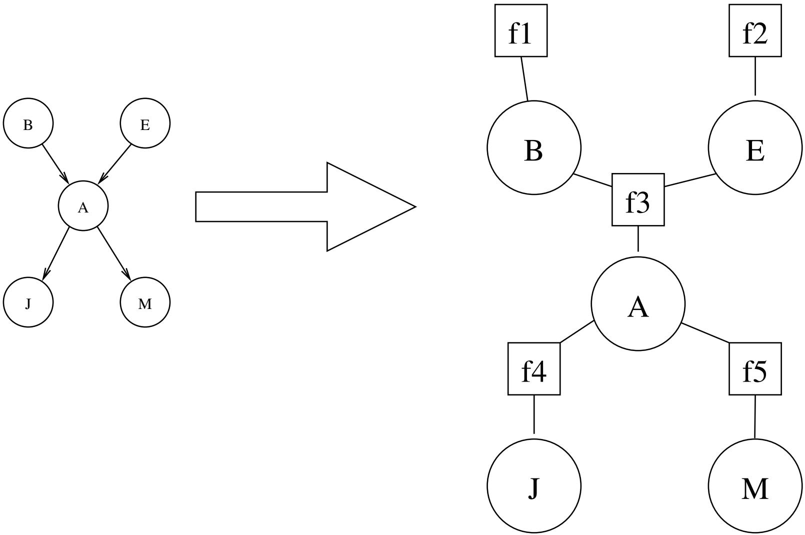
Factor Graph Example



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

Message Passing Ex.: Order of Computation



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

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