Recitation
Homework 3

10-418/618: Machine Learning for Structured Data

09/28/2022

Directed Graphical Models

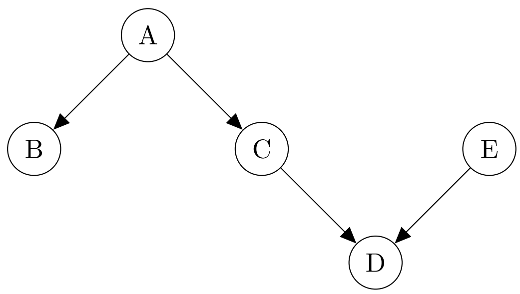
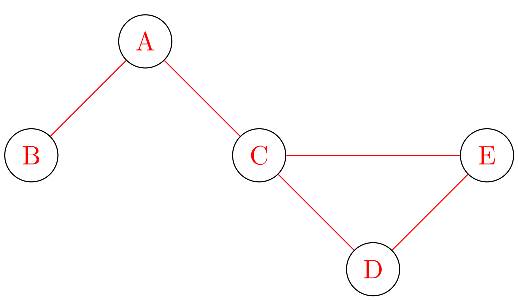


Figure 1: Directed Graphical Model

1. Consider the graph $\mathcal{G}$ given in Figure 1. Suppose you are given a joint distribution $P(A, B, C, D, E)$ and you are informed that P factorizes according to $\mathcal{G}$. Write down a factorization of P based on the definition of a directed graphical model.

$$P(A)\ P(B \mid A)\ P(C \mid A)\ P(E)\ P(D \mid C, E)$$

2. Draw the moralized graph of $\mathcal{G}$



Undirected Graphical Models

3. Identify the cliques in Fig. 2.

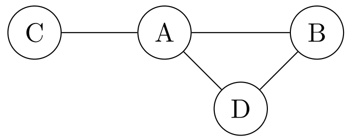


Figure 2: Undirected Graphical Model

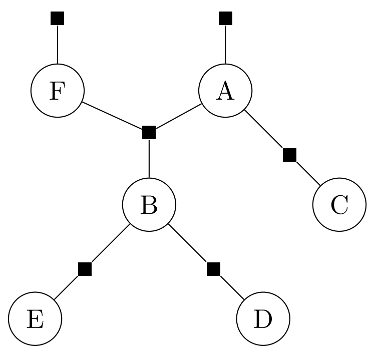
- $\{B, A, D\}$
- $\{A, C\}, \{A, D\}, \{A, B\}, \{B, D\}$
- $\{A\}, \{B\}, \{C\}, \{D\}$

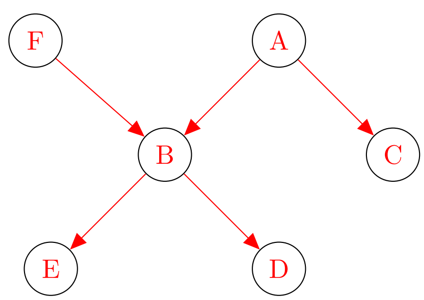
4. What is the Markov boundary of A? of D?

   $\{B, C, D\}$, $\{A, B\}$

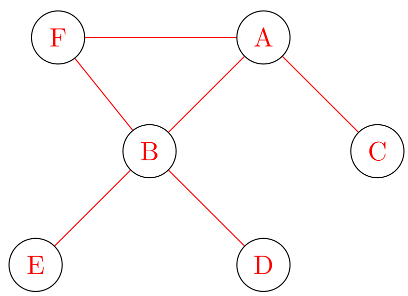
Factor Graphs

5. Convert the following factor graph to a directed graphical model and an undirected graphical model:





Directed GM



Undirected GM

Variable Elimination

6. When Querying $P(A|h = \tilde{h})$, Perform variable elimination on the following directed graph G in the order: H,G,F,E,D,C,B

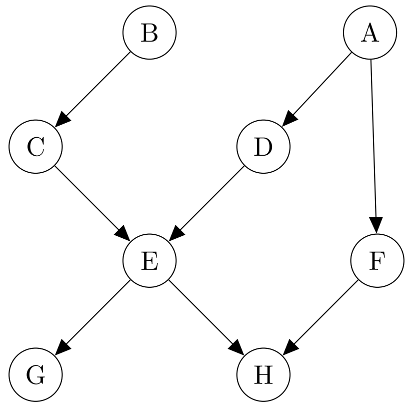


Figure 3: Initial graph for variable elimination

Variable Eliminated	Factor Computed
H	
G	
F	
E	
D	
C	
B	

Initial factorization:

$$P(a)P(b)P(c|b)P(d|a)P(e|c,d)P(f|a)P(g|e)P(h|e,f)$$

Variable Eliminated	Factor Computed
H	$m_h(e,f) = \sum_h p(h|e,f)\delta(h = \tilde{h})$
G	$m_g(e) = \sum_g p(g|e) = 1$
F	$m_f(e,a) = \sum_f p(f|a)m_h(e,f)$
E	$m_e(a,c,d) = \sum_e p(e|c,d)m_g(e)m_f(a,e)$
D	$m_d(a,c) = \sum_d p(d|a)m_e(a,c,d)$
C	$m_c(a,b) = \sum_c p(c|b)m_d(a,c)$
B	$m_b(a) = \sum_b p(b)m_c(a,b)$

7. Perform variable elimination on undirected graph G':

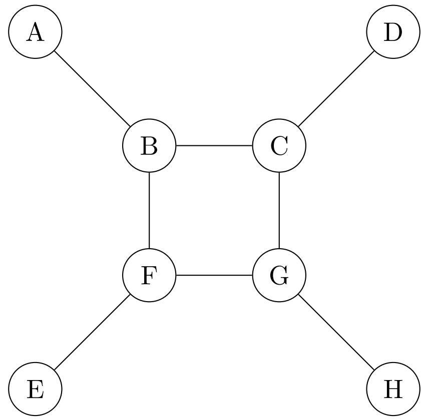


Figure 4: Initial graph for variable elimination

Order F, B, C, G, A, D, H, E:

Variable Eliminated	**Elimination Clique**
F	
B	
C	
G	
A	
D	
H	
E	

Order A, D, E, H, B, C, F, G:

Variable Eliminated	**Elimination Clique**
A	
D	
E	
H	
B	
C	
F	
G	

Order F, B, C, G, A, D, H, E:

Variable Eliminated	**Elimination Clique**
F	B, F, G, E
B	A, C, B, G, E
C	A, D, C, G, E
G	A, D, G, H, E
A	A, D, H, E
D	D, H, E
H	H, E
E	E

Order A, D, E, H, B, C, F, G:

Variable Eliminated	**Elimination Clique**
A	A, B
D	D, C
E	F, E
H	G, H
B	B, C, F
C	G, C, F
F	G, F
G	G

8. How does the size of the elimination cliques relate to the computational complexity of variable elimination?

   The time and space complexity are exponential in the size of the *largest* elimination clique.

Belief Propagation

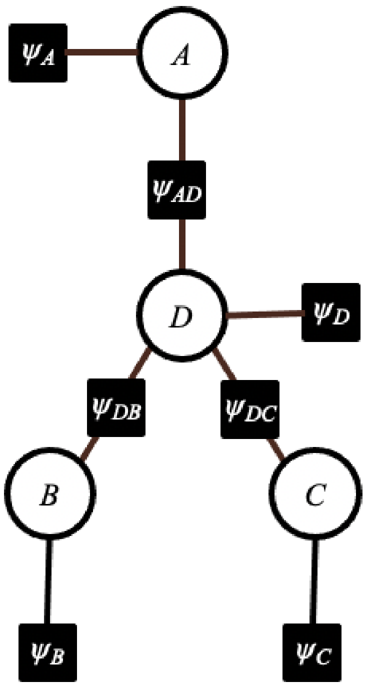


Figure 5

a	$\psi_A(a)$
0	3
1	2

b	$\psi_B(b)$
0	1
1	2

c	$\psi_C(c)$
0	3
1	1

d	$\psi_D(d)$
0	1
1	1

a	d	$\psi_{AD}(a,d)$
0	0	2
0	1	2
1	0	2
1	1	3

b	d	$\psi_{DB}(b,d)$
0	0	1
0	1	2
1	0	1
1	1	1

c	d	$\psi_{DC}(c,d)$
0	0	1
0	1	1
1	0	1
1	1	3

9. Consider the factor graph in Figure 5. On paper, carry out a run of belief propagation by sending messages first from the leaves ψ_B, ψ_C to the root ψ_A, and then from the root back to the leaves. Assume all messages are un-normalized. Then find the beliefs at nodes A, B, C, and D along with the beliefs for ψ_{DB}, ψ_{DC}, and ψ_{AD}.

upward messages:

$$\mu_{B\to\psi_{DB}} = \mu_{\psi_B\to B} = [1,2]^T$$

$$\mu_{C\to\psi_{DC}} = \mu_{\psi_C\to C} = [3,1]^T$$

$$\mu_{D\to\psi_{AD}} = \mu_{\psi_{DB}\to D}\mu_{\psi_{DC}\to D}\mu_{\psi_D\to D} = [3,4]^T[4,6]^T[1,1]^T = [12,24]^T$$

downward messages:

$$\mu_{A\to\psi_{AD}} = \mu_{\psi_A\to A} = [3,2]^T$$

$$\mu_{D\to\psi_{DB}} = \mu_{\psi_{AD}\to D}\mu_{\psi_D\to D}\mu_{\psi_{DC}\to D} = [10,12]^T[1,1]^T[4,6]^T = [40,72]^T$$

$$\mu_{D\to\psi_{DC}} = \mu_{\psi_{AD}\to D}\mu_{\psi_D\to D}\mu_{\psi_{DB}\to D} = [10,12]^T[1,1]^T[3,4]^T = [30,48]^T$$

beliefs:

$$b_B = \mu_{\psi_B\to B}\mu_{\psi_{DB}\to B} = [1,2]^T[184,112]^T = [184,224]^T$$

$$b_C = \mu_{\psi_C\to C}\mu_{\psi_{DC}\to C} = [3,1]^T[78,174]^T = [234,174]^T$$

$$b_D = \mu_{\psi_D\to D}\mu_{\psi_{DC}\to D}\mu_{\psi_{DB}\to D}\mu_{\psi_{AD}\to D} = [1,1]^T[4,6]^T[3,4]^T[10,12]^T = [120,288]^T$$

$$b_A = \mu_{\psi_A\to A}\mu_{\psi_{AD}\to A} = [3,2]^T[72,96]^T = [216,192]^T$$

b	d	$b_{\psi_{DB}}(b,d)$
0	0	40
0	1	144
1	0	80
1	1	144

c	d	$b_{\psi_{DC}}(c,d)$
0	0	90
0	1	144
1	0	30
1	1	144

a	d	$b_{\psi_{AD}}(a,d)$
0	0	72
0	1	144
1	0	48
1	1	144

Constituency Parsing

First we'll consider an extremely ambiguous sentence and see how we could disambiguate the meaning.

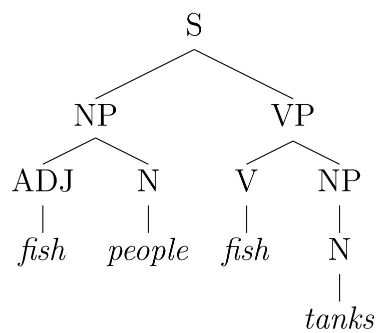
$$\text{Fish people fish tanks.} \tag{1}$$

This sentence could be interpreted two ways.

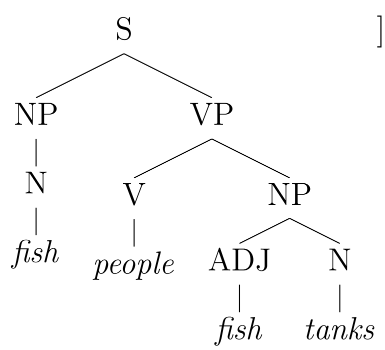
1. Fish-People hybrids fish in tanks.
2. Fish populate fish tanks. (This uses a less common meaning of people meaning "to populate".)

How could we build in structure to distinguish between these two meanings? One way is to build and label a parse tree.

1. Fish-People hybrids fish in tanks.



2. Fish populate fish tanks.



Both of these options can be summarized by a simple set of rules called a *grammar*. If you've taken a compiler course, you'll know and hate this term.

$$\begin{aligned} \textbf{S} &\rightarrow \textbf{NP VP} \\ \textbf{NP} &\rightarrow \textbf{ADJ N} \mid \textbf{N} \\ \textbf{VP} &\rightarrow \textbf{V NP} \end{aligned}$$

The takeaway from this is that if we could somehow generate these tree-labelings, we would be much closed to pulling *meaning* from a sequence of words before further processing. For example, this can be a useful first step in a dialogue system where disambiguating meaning is crucial.

So now that we've established the context for our problem of interest, let's set it aside for a little while.

Message Passing Review

Remember that for belief propagation on factor graphs, we have two types of messages we can pass.

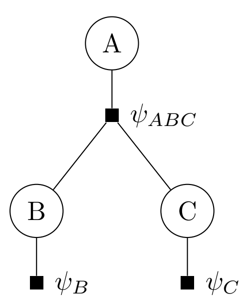
Variable to Factor:

$$\mu_{x\rightarrow f}(x) = \prod_{g\in \text{Ne}(x)/f} \mu_{g\rightarrow x}(x)$$

Factor to Variable:

$$\mu_{f\rightarrow x}(x) = \sum_{\mathcal{X}_f/x} \psi_f(\mathcal{X}_f) \prod_{y\in\{\text{Ne}(f)/x\}} \mu_{y\rightarrow f}(y)$$

We'll consider applying these rules to the following undirected graphical model.



Representation

First, let's consider the case where each potential function is tabular, variables B and C are binary, and variable A has N possible settings.

Q: How would we represent this as a NumPy array?

A: Use an array where each axis i represents all the values variable i can take.

Q: What would be the size of ψ_b?

A: $(2,)$

Q: What would be the size of ψ_{abc}?

A: $(N, 2, 2)$

Q: How would we marginalize out a variable in a potential function in this scenario?

A: Sum over the axis that represents the variabel you want to marginalize.

Q: How would we express each of the types of messages in code?

A: This is a straightforward combination of element wise multiplication and summing over your axis of interest.