Graphical models

Review

- Graphical models (Bayes nets, Markov random fields, factor graphs)
  - graphical tests for conditional independence (e.g., d-separation for Bayes nets; Markov blanket)
  - format conversions: always possible, may lose info
  - learning (fully-observed case)
- Inference
  - variable elimination
  - today: belief propagation

Junction tree

(aka clique tree, aka join tree)

- Represents the tables that we build during elimination
  - many JTs for each graphical model
  - many-to-many correspondence w/ elimination orders
- A junction tree for a model is:
  - a tree
  - whose nodes are sets of variables ("cliques")
  - that contains a node for each of our factors
  - that satisfies ***running intersection property***

nodes are cliques:
    these are the tables we build

a node for each factor:
    factor is a subset of that node's clique

Running intersection property

- In variable elimination: once a variable X is added to our current table T, it stays in T until eliminated, then never appears again
- In JT, this means all sets containing X form a connected region of tree
  - true for all X = running intersection property

Incorporating evidence (conditioning)

- For each factor or CPT:
  - fix known arguments
  - assign to some clique containing all non-fixed arguments
  - drop observed variables from the JT
- No difference from inference w/o evidence
  - we just get a junction tree over fewer variables
  - easy to check that it's still a valid JT

Message passing (aka BP)

- Build a junction tree (started last time)
- Instantiate evidence, pass messages (calibrate), read off answer, eliminate nuisance variables
- Main questions
  - how expensive? (what tables?)
  - what does a message represent?

what does a message represent?

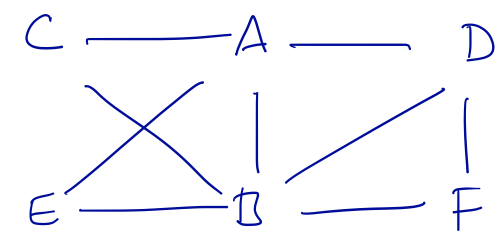
all of the information from a region of the junction tree: a single small potential that serves as a surrogate for a larger portion of tree

the result of variable elimination: if we marginalized away all variables in part of the tree, and if the only potentials were the ones that have just these variables as arguments, message would be the resulting marginal

Example

CEABDF

	S	T	V
0	∅		
1	AB	ABC	C
2	AB	ABE	CE
3	BD	ABD	CEA
4	DF	BDF	CEAB
5	F	DF	
6	∅	F	



$$\sum_A\sum_B\sum_C\sum_D\sum_E\sum_F \phi_1(ABC)\,\phi_2(ABE)\,\phi_3(ABD)\,\phi_4(BDF)$$

$$\sum_F\sum_D\sum_B \phi_4(BDF)\sum_A \phi_3(ABD)\sum_E \phi_2(ABE)\sum_C \phi_1(ABC)$$

$$\sum_F\sum_D\sum_B \phi_4(BDF)\sum_A \phi_3(ABD)\sum_E \phi_2(ABE)\sum_C \phi_1(ABC)$$

elimination order turns first expression into 2nd

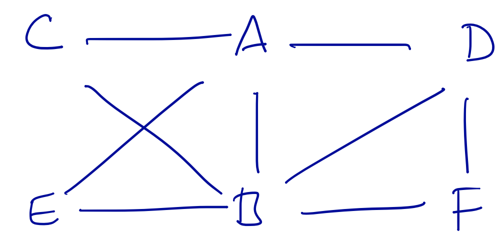
parens around inner 3 sums: these represent small tables that we compute and later multiply into some larger table — “messages”

one for each local min of S: AB, AB, BD (none for DF since F and null set immediately follow and are subsets) — psi_1(AB), psi_2(AB), psi_3(BD)

or we could re-parenthesize: psi_1 could be either inside sum over E or (since E is not an argument) it can be factored out

difference between multiplying psi_1 into phi_2 or phi_3 — slightly different junction trees

What if order were FDBAEC?



would still create cliques BDF, ABD, ABE, ABC (same but in reverse order)

would still get same junction tree(s), but now messages pass in reverse direction — e.g., summing D out of ABD gives a message over AB that we later multiply into ABC

in general, many elimination orders can lead to same junction tree; messages could pass in either direction over an edge depending which side of the edge gets summed out first.

Messages

- Message = smaller tables that we create by summing out some variables from a clique
  - we later multiply the message into exactly one other clique before summing out that clique
  - one message per edge (e.g., ABC — ABD)
  - arguments of message: intersection of endpoints (AB)
    - called a ***sepset*** or separating set
  - message might go in either direction over the edge depending on which side of the JT we sum out first

Interesting fact: fix an edge in the JT and a direction; then no matter how we order the eliminations (consistent with this JT and edge direction), the message over this edge will be the same

to see why: we've eliminated all variables that appear only on one side of the tree, and none that appear on the other side, so the order of eliminations didn't matter

Belief propagation

- Idea: calculate all messages that could be passed by any elimination order consistent with our JT
- For each edge, need two runs of variable elimination: one using the edge in each direction
- Insight: that's just two runs total

Belief propagation

- Pick a node of JT as root arbitrarily
- Run variable elimination outward from the root
  - any order is OK as long as we do edges closer to the root first
- Run variable elimination inward toward the root
- Done!
  - passed one message in each direction over each edge

All for the price of two

- Now we can simulate any order of elimination consistent with the tree:
  - orient JT edges in the direction consistent with the elimination order
  - these are the messages that elimination would compute

Example

Tree: AB — BC — CD, BC — CE — EF
Potentials: all [2 1 1 2] for [TT TF FT FF]
Observe: D = true (so CD potential becomes [2 1])
pick AB as root
messages root -> leaves
AB -> BC: B, [3 3]
BC -> C: C, [9 9]
BC -> CE: C, [9 9]
CE -> EF: E, [27 27]
messages leaves -> root
C -> BC: C, [2 1]
EF -> CE: E, [3 3]
CE -> BC: C, [9 9]
BC -> AB:
9 * [2 1 1 2] .* [2 1 2 1] = 9*[4 1 2 2] => [45 36]

Using it

- Want: P(B, C | D=T)
  - i.e.,
- Variable elimination:

sum_A sum_E P(ABCE | D=T)
sum_A sum_E ~P(ABCE | D=T) /
  sum_ABCE ~P(ABCE | D=T)

where ~P is unnormalized probability (product of all potentials)

i.e., two runs of variable elimination: one to eliminate AE, one to eliminate everything

to elim AE: we use messages in from AB (const 3) and CE (const 27), potential at BC ([2 1 1 2]), and trivial message from CD ([2 1])
mult all together: 81 * [4 1 2 2]
norm: [4 1 2 2]/9

Marginals

- More generally, marginal over any subtree:
  - product of all incoming messages and all local factors
  - normalize
- Special case: clique marginals

Read off answer

- Find some subtree that mentions all variables of interest
- Compute distribution over variables mentioned in this subtree
  - product of all messages into subtree and all factors inside subtree / normalizing constant
- Marginalize (sum out) nuisance variables

depending on query and JT, might have a lot of nuisance variables

Inference—recap

- Build junction tree (e.g., by looking at tables built for a particular elimination order)
- Instantiate evidence
- Pass messages
- Pick a subtree containing desired variables, read off its distribution, and sum out nuisance variables

Calibration

- After BP, easy to get all clique marginals
  - also all sepset marginals (sum out from clique on either side)
- Bayes rule: P(clique \ sepset | sepset) =
- So, joint P($\text{clique}_1 \cup \text{clique}_2$) =
- Continue over entire tree: P(everything) =

Bayes: … = P(clique) / P(sepset)
joint: P(c1) P(c2 | c1) = P(c1) P(c2 | sepset)
 = P(c1) P(c2) / P(sepset)

P(everything) = prod P(clique_i) / prod P(sepset_j)

calibrated JT: one where we know all clique and sepset marginals

Hard v. soft factors

Hard

	X = 0	X = 1	X = 2
Y = 0	0	0	0
Y = 1	0	0	1
Y = 2	0	1	1

Soft

	X = 0	X = 1	X = 2
Y = 0	1	1	1
Y = 1	1	1	3
Y = 2	1	3	3

number = degree to which event is more or less likely
must be nonnegative

0 = hard constraint

can combine hard & soft (some numbers zero, others positive and varying)

hard factors can lead to complications (e.g., impossible to satisfy all constraints; e.g., Koller ex 4.4 (may not be able to factor according to a graph that matches our actual set of independences, i.e., failure of Hammersley-Clifford))

we'll mostly be using soft factors

Moralize & triangulate (to build JT)

- Moralize:
  - for factor graphs: a clique for every factor
  - for Bayes nets: "marry the parents" of each node
- Triangulate: find a chordless 4-or-more-cycle, add a chord, repeat
- Find all maximal cliques
- Connect maximal cliques w/ edges in any way that satisfies RIP

Connect maximal cliques w/ edges in any way that satisfies RIP (NP-hard to find best way, but any elimination order yields one)

Loopy BP

Plate models

you've seen one already: naive Bayes

a typical example: LDA

other macro languages: MLNs, ICL