BN Semantic II
d-Separation, PDAGs, etc

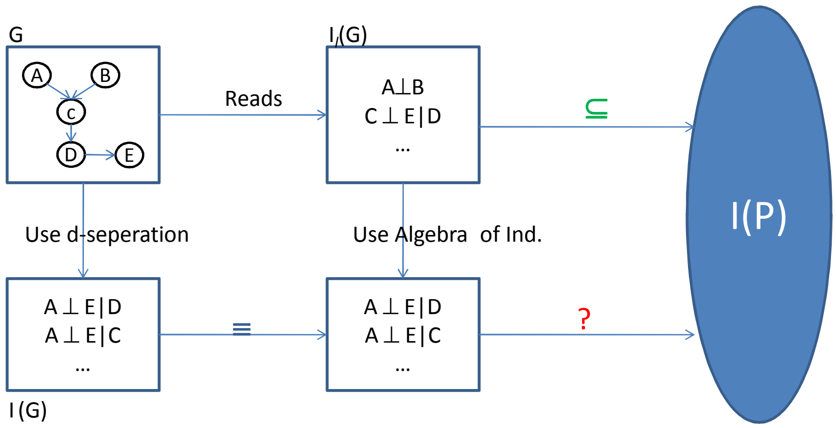
Amr Ahmed

Sep 25th, 2008

Outline

- Independence
- D-separation and Active Trails
- P-dag learning

Independence



-If G is an I-map of P then $I_l(G) \subseteq I(P)$
- Also, it is always true that $I(G) \subseteq I(P)$ means d-separation is **sound**
- And for almost all Ps that factor over G, I(G) = I(P) , P is faithful to G

D-Separation

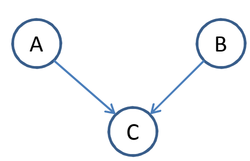
- A graph algorithm for answering independence queries over G
- Sound and complete for almost all Ps that factor according to G
- P is faithful if it doesn't declare extra independence assumption that can't be read from G

False	True
0.8	0.2

False	True
0.8	0.2

P entails C ⊥ A,B !!

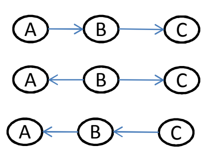
P is not faithful



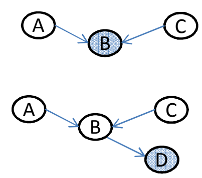
AB\C	False	True
FF	0.9	0.1
FT	0.9	0.1
TF	0.9	0.1
TT	0.9	0.1

D-Separation Cont

- Q: is X⊥ Y | Z?
- Answer by contradiction
  - Find a way that information flows between X and Y despite the existence of Z
  - Information can flow if there is a path x…y that is not blocked by z **(active trail)**
- **Very simple "local" rules**



Can move from A to C if B is not in Z



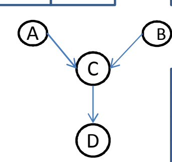
Can move from A to C if B is in Z or one of its decedents

Understanding the V-structure more

-C is a noisy X-or of A and B
- If C is not observed then A and B are uniform
- If you observe C, then A and B are dependent
  -C=1, ➔ A= not B w.h.p
  -C=0 → A=B w.h.p

False	True
0.5	0.5

False	True
0.5	0.5



AB\C	F	T
FF	0.9	0.1
FT	0.1	0.9
TF	0.1	0.9
TT	0.9	0.1

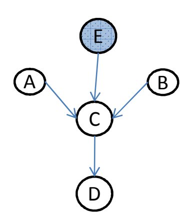
Observing a decedent

-D is a noisy NOT of C
- If you observe D, then w.h.p you know C
-If you have an idea about C, A and B are dependent

C\D	False	True
False	.1	0.9
True	0.9	0.1

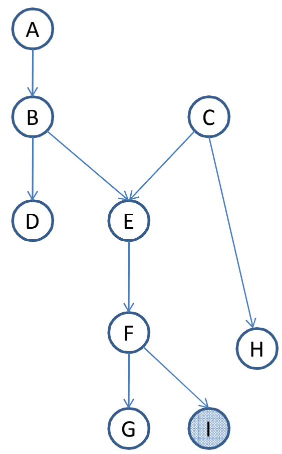
Observing an ancestor!!

-Won't help, E alone shouldn't tell us much about C
- But in some situations it DOES, but later in the semester (context-specific independence)



D-separation Example

-Given I is A ⊥ C
- Given I is A ⊥ F
- Given I and B is A ⊥ C



Why D-separation is useful?

- Intuitively and on an abstract level, when you answer a probabilistic query P(A|B), you would like to consider only those variables that would affect A given B
- Later, when we talk about inference, we will visit this again
- The concept of active trails is really very important in proving and justifying algorithms
  - You **should** use it in Q2 and Q4

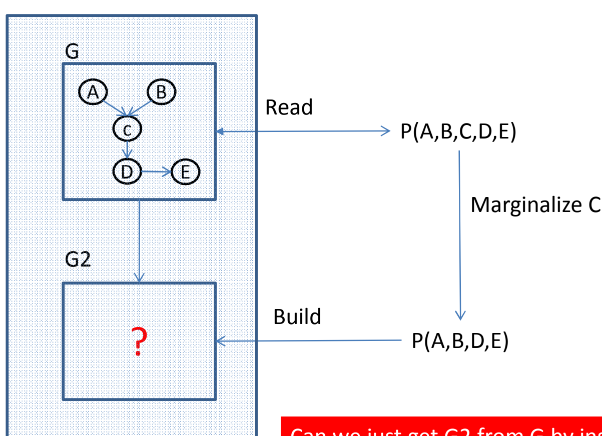
Active Trails

- If it is all about independence, then to show that two graphs, G1 and G2 are equivalent, we need to show that I(G1) = I(G2), or practically:
  - A trail is active in G1 iff it is active in G2**
  - More algorithmically
    - Consider all ways in which some of the variables are observed
    - Show that all active trails in G1 and G2 are the same

** This is only true if G1 and G2 have no triangles, in case of triangles, we require that they agree on the set of **minimal active trails** (see problem 3.16). For this homework, we won't worry too much about this subtlety.

Question 4 again

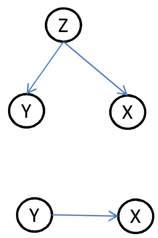
- Marginalization is a key operation, that we will use later in the semester.

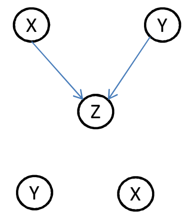


Can we just get G2 from G by inspecting the graph?

Simple Marginalization

We can do it graphically





We can also do it algebraically

Chain rule

$$P(X,Y)=\sum_z P(X,Y,z)$$
$$=\sum_z P(X)P(Y\mid X)P(z\mid X,Y)$$
$$=P(X)P(Y\mid X)\sum_z P(z\mid X,Y)$$
$$=P(X)P(Y\mid X)$$

Marginally dependent

Factorize as in G

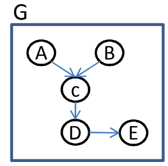
$$P(X,Y)=\sum_z P(X,Y,z)$$
$$=\sum_z P(X)P(Y)P(z\mid X,Y)$$
$$=P(X)P(Y)\sum_z P(z\mid X,Y)$$
$$=P(X)P(Y)$$

Marginally independent

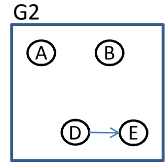
Q4 again

- Removing C, introduces new independence assumptions not in G like A ⊥ D
- **We need to add more edges to compensate**
- You need to consider what active trails are enabled by C
  - A → C → D
  - But also, A→C ←B given D?
- Think what need to be done to make sure that the end variables are still dependent in G2 under the same conditions but when C is marginalized
- Sometimes fixing a trail will fix the other ones (we need to add the minimum number of edges)
- Hint: think first about trails that don't require observing any other variables as fixing them might fix the others!

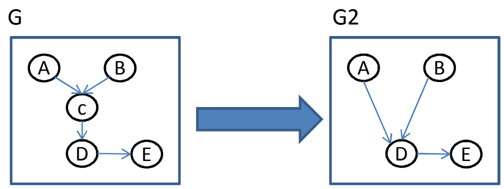
G

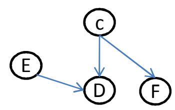


G2

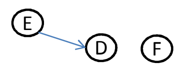


Q4





?



D←C→F
E→D←C→F given D

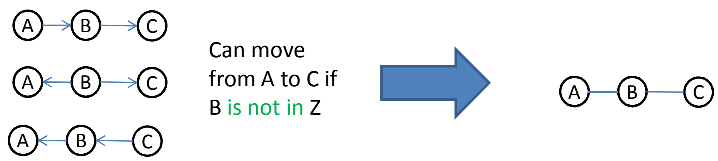
How to get these dependencies right after removing C?
(the above graph is not the solution!!, You should think about it)

Outline

- Independence
- D-separation and Active Trails
- P-dag learning

PDAG

- PDAG is a compact way of representing equivalent graphs
- Orient edges only if they must be this way
- Undirected edges can be either way
  - Remember key is active trails
  - For some active trails (other than v-structure --- immoralities--) edge direction is not important



Learning P-DAGs

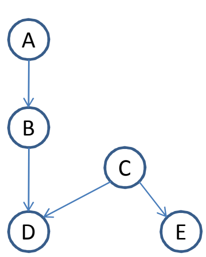
- Learning the skeleton
- Discovering immoralities
- Orienting edges (this is straightforward)

Learning the skeleton

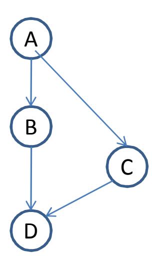
- There is an edge between X,Y if you can not stop information flow between them
- You can stop information flow if you can block all paths between X,Y
- You can **block a path**, if you observe some variables (possibly empty set) = U
- The test:
  - Can you find U such that **X $\perp$ Y | U**?
    - If NO → then x—Y
    - If Yes → then there is no edge

Step 1: Learning the skeleton

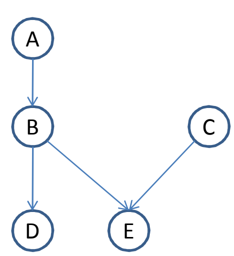
- Test: Can you find U such that **X $\perp$ Y | U**?
  - What is U? subset of all variables- {X,Y}
  - Can go up to size d (max fan in, or degree)? Why?
  - You don't have to go over all possible U
    - A witness is all what you need to answer YES



What is a witness for E,D?



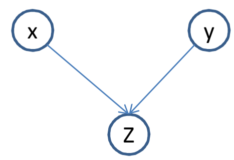
What is a witness for A,D?



What is a witness for B,C?

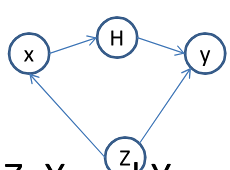
Step 2: Discover Immoralities

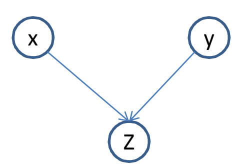
- For immoralities, we must direct edges in a certain way, so we should discover them
- A v-structure with no married parents
- Simple test:
  - Is X dependent on Y given Z?
    - If yes, …
    - If no, …



Step 2: Discover Immoralities

- This simple test will introduce false positives
- If it is a true immorality, we are OK
- But what about:

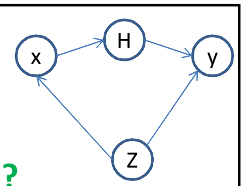




- Given Z, X and Y are dependent
  - But not via Z, unfo. via another path

Step 2: Discover Immoralities

- Simple test: Is X dependent on Y given Z? that fails
- Should be: Is X dependent on Y given Z via a path that goes only through Z?
  - Practically we should **block** all other paths that lead from X to Y
  - In addition to observing Z, we might observe as many other variables as possible
- Test: Is X dependent on Y for all U, z in U?
  - Yes→ immorality
  - NO→ not immorality



- Answer is No, witness U= {Z,H}
- We are really asking the same questions
- Skeleton: Can you find U such that **X ⊥ Y | U?**
  - **Yes → no edge,**
  - **No → an edge**
- **Immorality**: Can you find U such that **X ⊥ Y | U?**
  - **Yes, and z in U → not immorality**
  - **Yes, and z not in U → immorality**
  - **The answer here can NOT be NO, why?**
- This has been exploited via cashing in the book (but see the extra credit problem)
  - As instructed, you **shouldn't** cache in your solution.
  - You should consider all U that **contains Z** until you find a witness (if there is one)

Some Hints to the programming problem

- A suggestion about representing PDAG
  - G(a,b)= 1 , G(b,a) = 1 if a—b
  - G(a,b) = 2 and G(b,a) = 0 if a→b
  - G(a,b) = 0 and G(b,a) = 2 if a←b
  - Makes life easier, ex. Check if a→b
    - If (G(a,b) ==2)
    - In old representation, if(G(a,b)==1 && G(b,a)==0)
- Size of U in witness test
  - You need only up to d
  - But it won't hurt to go up to 2*d, why?
    - After all you are looking for a witness.