University of Washington
Department of Electrical Engineering
EE512 Spring, 2006
Graphical Models

Jeff A. Bilmes <bilmes@ee.washington.edu>

Lecture 3 Slides
April 4th, 2006
Outline of Today’s Lecture

• Start computing probabilities
• elimination algorithm
• Beginning of chordal graph theory
Books and Sources for Today

- Jordan: Chapters 2 and 3
- Any graph theory text.
## Class Road Map

| L1: Tues, 3/28 | Overview, GMs, Intro BNs |
| L2: Thur, 3/30 | semantics of BNs + UGMs |
| L3: Tues, 4/4  | elimination, probs, chordal I |
| L4: Thur, 4/6  | |
| L5: Tue, 4/11 | |
| L6: Thur, 4/13 | |
| L7: Tues, 4/18 | |
| L8: Thur, 4/20 | |
| L9: Tue, 4/25  | |
| L10: Thur, 4/27 | |

| L11: Tues, 5/2  | |
| L12: Thur, 5/4  | |
| L13: Tues, 5/9  | |
| L14: Thur, 5/11 | |
| L15: Tue, 5/16  | |
| L16: Thur, 5/18 | |
| L17: Tues, 5/23 | |
| L18: Thur, 5/25 | |
| L19: Tue, 5/30  | final presentations |
| L20: Thur, 6/1  | |

Lec 3: April 4th, 2006
EE512 - Graphical Models - J. Bilmes
Announcements

• If you see a typo, please tell me during lecture
  – everyone will then benefit.
  – note, corrected slides will go on web.

• READING: Chapter 3 in Jordan’s book (take a look at chapters 17)

• Please audit if you are attending class
  – it will help to get us a bigger room.
  – If you need an add code, just send me email.

• Reminder: TA discussions and office hours:
  – Office hours: Thursdays 3:30-4:30, Sieg Ground Floor Tutorial Center
  – Discussion Sections: Fridays 9:30-10:30, Sieg Ground Floor Tutorial Center Lecture Room
Computing Probabilities

- Typical goal. Given distribution over \( N \) variables, \( V = \{1, \ldots, N\} \), \( F \cup E \cup H = V \), \( F, E, H \) disjoint, compute: \( p(x_F, x_E) = \sum_{x_H} p(x_F, x_E, X_H) \).
- Complexity: \( O(r^N) \), we do \( r \) additions \( r^{N-1} \) times, true even if \( H = \{j\} \), just one index.
- Factorization can help: exploit the local structure to reduce computation.

\[
p(x_{1:6}) = p(x_1)p(x_2|x_1)p(x_3|x_1) \cdot p(x_4|x_2)p(x_5|x_3)p(x_6|x_2, x_5)
\]

\[
p(x_{1:5}) = \sum_{x_6} p(x_1)p(x_2|x_1)p(x_3|x_1)p(x_4|x_2)p(x_5|x_3)p(x_6|x_2, x_5)
\]

\[
= p(x_1)p(x_2|x_1)p(x_3|x_1)p(x_4|x_2)p(x_5|x_3) \sum_{x_6} p(x_6|x_2, x_5)
\]

\[
= p(x_1)p(x_2|x_1)p(x_3|x_1)p(x_4|x_2)p(x_5|x_3) \quad O(r^6) \text{ ops.}
\]
Computing Probabilities

- Only $r^3$ calculations, by bringing sum into factors using distributive law of arithmetic, i.e., $ab + ac = a(b + c)$, or more generally:

$$\sum_{i_1} \sum_{i_2} \ldots \sum_{i_N} f(i_1)f(i_2) \ldots f(i_N) = \left( \sum_{i_1} f(i_1) \right) \left( \sum_{i_2} f(i_2) \right) \ldots \left( \sum_{i_N} f(i_N) \right)$$

- We will use this property to compute $p(x_1 | \bar{x}_6) = p(x_1, \bar{x}_6)/p(\bar{x}_6)$ where it is assumed that $x_6$ is given (thus the bar notation).

- Generic notation of summing out a variable. $\phi_{X_i}(x_A) = \sum_{x_i} f(x_i, x_A)$

$$p(x_1, \bar{x}_6) = \sum_{x_{2:6}} p(x_1)p(x_2|x_1)p(x_3|x_1)p(x_4|x_2)p(x_5|x_3)p(x_6|x_2, x_5) \quad \text{O}(r^3) \text{ ops.}$$

$$= p(x_1) \sum_{x_2} p(x_2|x_1) \sum_{x_3} p(x_3|x_1) \sum_{x_4} p(x_4|x_2) \sum_{x_5} p(x_5|x_3) \sum_{x_6} p(x_6|x_2, x_5)$$

$$= p(x_1) \sum_{x_2} p(x_2|x_1) \sum_{x_3} p(x_3|x_1) \sum_{x_4} p(x_4|x_2) \sum_{x_5} p(x_5|x_3) \phi_{X_6}(x_2, x_5)$$
Computing Probabilities

\[ p(x_1) \sum_{x_2} p(x_2 | x_1) \sum_{x_3} p(x_3 | x_1) \sum_{x_4} p(x_4 | x_2) \sum_{x_5} p(x_5 | x_3) \phi_{X_6}(x_2, x_5) \]

\[ = p(x_1) \sum_{x_2} p(x_2 | x_1) \sum_{x_3} p(x_3 | x_1) \sum_{x_4} p(x_4 | x_2) \phi_{X_5}(x_2, x_3) \]

\[ = p(x_1) \sum_{x_2} p(x_2 | x_1) \sum_{x_3} p(x_3 | x_1) \phi_{X_5}(x_2, x_3) \sum_{x_4} p(x_4 | x_2) \]

\[ = p(x_1) \sum_{x_2} p(x_2 | x_1) \phi_{X_4}(x_2) \sum_{x_3} p(x_3 | x_1) \phi_{X_5}(x_2, x_3) \]

\[ = p(x_1) \sum_{x_2} p(x_2 | x_1) \phi_{X_3}(x_1, x_2) \phi_{X_4}(x_2) \]

\[ = p(x_1) \phi_{X_2}(x_1) \]

\[ = p(x_1, \bar{x}_6) \]

And we can finish this off with:

\[ \phi_{X_1} = p(\bar{x}_6) \]

\[ p(x_1 | \bar{x}_6) = \frac{p(x_1) \phi_{X_2}(x_1)}{\phi_{X_1}} \]

Total computation is only \( O(r^3) \) ops.
Note on sums and evidence

- In the summation form, each sum over \( x_i \) was over only those factors that involved variable \( x_i \), which helped to minimize computation.

- Evidence: some random variables are observed, others are hidden. "evidence" nodes are ones that we know the value of (it is not that they have probability of 1).

- Graphically, we typically shade in evidence nodes.

- For a particular value of evidence \( \bar{x}_E \), we write

\[
p(x_F|x_E)\delta(x_E, \bar{x}_E) = \begin{cases} 
p(x_F|\bar{x}_E) & \text{if } x_E = \bar{x}_E \\ 
0 & \text{else} \end{cases}
\]

- So, \( p(x_1, \bar{x}_6) = \sum_{x_6} p(x_1, x_6)\delta(x_6, \bar{x}_6) \), or any factor not conforming to evidence is multiplied by 0.

- Note, mathematically, we’re just summing a collection of factors:

\[
p(\bar{x}_E) = \sum_{x_F} p(x_F, \bar{x}_E)
\]
Graph Algorithm Equivalent: Elimination

- There is a purely graph-theoretic algorithm on undirected graphs called *elimination* (or variable elimination).
  1) Choose an elimination order of the variables (called $I$)
  2) Choose next node $v$ in the elimination order and connect all neighbors in the current graph (after this step, all neighbors of $v$ are now neighbors themselves).
  3) Remove (eliminate) $v$ and all its edges from the graph, goto 2.

- Ex: $I = (6, 5, 4, 3, 2, 1)$. 

![Graph Algorithm Equivalent: Elimination Diagram]
Elimination Example: different graph

Order the Nodes
Eliminate the nodes in order
Result of Elimination

- Reconstituted graph: is *triangulated* or *chordal*. It has important properties, we’ll be going over chordal graph theory. Note now, however, that this is same as decomposable models.
Elimination

- Elimination cliques: \( C_i = X_i \cup \{ \text{neighbors of } X_i \text{ at elimination time} \} \)
- Largest elimination clique determines the exponent of the complexity of inference. E.g., \( C_6 = \{6, 5, 2\} \), \( C_5 = \{2, 5, 3\} \), \( C_4 = \{2, 4\} \), \( C_3 = \{1, 2, 3\} \), \( C_2 = \{2, 1\} \), \( C_1 = \{1\} \).
- Ex: \( \phi_{x_3}(x_1, x_2) = \sum_{x_3} p(x_3|x_1)\phi_{x_5}(x_2, x_3) \), has clique of size 3, or \( O(r^3) \) operations.

- Goal: find an elimination ordering that minimizes the size of the largest elimination clique (to reduce the exponent of \( r \)). In other words, the largest elimination clique size we encounter in the graphical elimination algorithm is equal to the largest dimensionality of the table size (exponent of \( r \)) during summing.
Elimination

- Problem: finding the best elimination order is an NP-hard optimization problem (Arnborg 1978).

- Theorem: The following decision problem is NP-complete: Given a graph $G$ and an integer value $k$, decide if an elimination order exists such that the size of the largest elimination clique is $\leq k$. Reduces from Arnborg 1978.

- There are **many** heuristics (we’ll have more details later in the course when we discuss triangulated and chordal models) few with any approximation guarantees though.
Elimination & Directed Graphs

- What if we start with directed graph, where we have factors $p(x_i | x_{\pi_i})$, so each sum over a variable $x_i$ in such a factor will certainly involve all of its parents.
- When converting from BN to MRF, if we just drop arrow directions and run graphical elimination, we might never get elimination clique representing the BN’s local factors (i.e., the BN’s factors would not be respected and would be lost).
- To avoid this from happening, we connect all unconnected parents when moving to MRF from BN. Called ”moralization”.

![Diagram showing before and after moralization process]
Immoral Elimination

- If we didn’t moralize, we would run into trouble. Example, suppose we eliminate $x_5$ on a graph that has not been moralized.

- Elimination clique $C_5 = \{3, 6, 5\}$ wouldn’t be able to ”hold” the directed factor $p(x_6|x_2, x_5)$, nor would any future elimination clique.
- Equivalently, we can’t validly distribute variable $x_5$ to the right $x_2$ in the factor: $\ldots \sum_{x_5} p(x_6|x_2, x_5)p(x_5|x_3)$ due to the interaction of $x_6, x_2, x_5$ — these three are unfactorizable (in general).
- If we first moralized and then summed out $x_5$, we’d get interaction clique $C_5 = \{5, 3, 6, 2\}$ which is mathematically ok, but not computationally optimal $O(r^4)$.
Moralization

- Q: What is effect on conditional independence statements when we moralize? What happens to family?

- Each CPT (factor) in the BN becomes a clique in the MRF. Therefore, the MRF will always have a clique that can "hold" each original CPT. Also, unfactorizability (in general case) of CPT is preserved in MRF, so we don’t gain any factorization properties.

- But we lose factorization properties, namely in V-structures, we lose the independence of the unconnected parents. Does this hurt us computationally when doing elimination?
Variable Elimination Algorithm

- We get the "variable elimination" algorithm.
- First, if starting from a BN, moralize the graph to obtain a MRF.

Choose an ordering $I$;
Consider all factors as active;
while any factors remain do
  1. Choose next variable $v$ in ordering;
  2. Deactivate factors that involve $v$, multiply together to form $\phi$;
  3. Sum out $v$ from $\phi$;
  4. Activate new summed out factor;
end

- We notice that this numerical algorithm is exactly analogous to the graph theoretic elimination algorithm from slide 10.
Summary

- We have BNs and MRFs
- The elimination algorithm (variable elimination) is seen as either graph theoretic or mathematical, for computing $p(x_E)$ (i.e., inference), where $x_E$ is evidence (observed) nodes. Evidence can be seen as a hidden node but with a delta function within a summation.
- Moralization takes us from BN to MRF, and MRF is where the work gets done.
- NP-complete to find optimal way of doing inference (computing $p(x_E)$).
- What if we want to do more than $p(x_E)$? What about more general things such as $p(x_F|x_E)$ or $\arg \max_{x_F} p(x_F|x_E)$? What if we want to do this for all $F$ using, say, dynamic programming?
- To do this exactly, we need to understand the basics of chordal graph theory. To do it approximately, there are other methods.
More Graph Terminology/Notation

- Graph $G = (V, E)$, $V$ finite vertex set, and $E \subseteq (V \times V)$ ordered pair of vertices, or edges. No self loops.
- If $\forall(\alpha, \beta) \in E$, $(\beta, \alpha) \in E$, then $G$ is undirected, and $\alpha$ and $\beta$ are neighbors, written $\alpha \sim \beta$.
- If $(\alpha, \beta) \in E$ and $(\beta, \alpha) \notin E$, we have directed edge $\alpha \rightarrow \beta$, $\alpha$ is a parent of $\beta$. If $\alpha \rightarrow \beta$, then $\alpha$ is a parent of $\beta$ and $\beta$ is a child of $\alpha$. $pa(\beta)$ is the set of all parents of $\beta$, and $ch(\alpha)$ is the set of all children of $\alpha$.
- neighbors of $\beta$, $ne(\beta) = \{\alpha : \alpha \sim \beta\}$
- parents of a set $pa(A) = \bigcup_{\alpha \in A} pa(\alpha) \setminus A$
- neighbors of a set $ne(A) = \bigcup_{\alpha \in A} ne(\alpha) \setminus A$
- children of a set $ch(A) = \bigcup_{\alpha \in A} ch(\alpha) \setminus A$
- boundary of a set of vertices $bd(A) = pa(A) \cup ne(A)$
- closure of a set, $cl(A) = A \cup bd(A)$
- recall def of vertex induced subgraph (lecture 2, slide 23)
Graph Terminology/Notation

- **maxclique**: A clique $A \subseteq V$ is maximal (with respect to $\subseteq$) if adding any node $v \in V \setminus A$ to $A$ makes $A \cup \{v\}$ not complete.

Maxcliques: \{E, \{E, F\}, \{F, G\}, \{G, B\}\}

- **path**: of length $n$ from $\alpha$ to $\beta$, seq. $\alpha = \alpha_0, \alpha_1, \ldots, \alpha_n = \beta$ of distinct vertices s.t. $(\alpha_{i-1}, \alpha_i) \in E, \forall i$ (paths do not go against edge directions)
  - **directed path**: if $\exists i : \alpha_{i-1} \rightarrow \alpha_i$
  - If $\exists$ path $\alpha$ to $\beta$, $\alpha \rightarrow \beta$
  - $\alpha, \beta$ connect if $\alpha \leftrightarrow \beta$ and $\beta \leftrightarrow \alpha$, or $\alpha \equiv \beta$.
  - Note: equivalence relation (class), $\alpha \equiv \beta$ means $\beta \in [\alpha]$, the strong components of $\alpha$.
• trail from $\alpha$ to $\beta$ is like a path, but arrow directions don’t matter.
• well (or topological) ordering: of nodes in DAG: A numbering (bijection) $\# : V \leftrightarrow \{1, \ldots, |V|\}$ such that if $\alpha \to \beta$ then $\#(\alpha) < \#(\beta)$ (we also speak of $\#^{-1}$)
  • predecessors: $\text{pr}_\#(\alpha) = \{\beta : \#(\beta) < \#(\alpha)\}$
  • ancestors: of $\beta$, $\text{an}(\beta) = \{\alpha : \alpha \leftrightarrow \beta \text { but not } \beta \leftrightarrow \alpha\}$
  • descendants: of $\alpha$, $\text{de}(\alpha) = \{\beta : \alpha \leftrightarrow \beta \text { but not } \beta \leftrightarrow \alpha\}$
  • non-descendants: $\text{nd}(\alpha) = V \setminus (\alpha \cup \text{de}(\alpha))$

$$\text{nd}(D) = \{A, B, C, E\}$$
$$\text{nd}(E) = \{A, B, C, D\}$$
$$\text{nd}(F) = \{A, B, C, D, E\}$$
Graph Terminology/Notation

- **Ancestral set**: $A \subseteq V$ is ancestral set if $\text{bd}(\alpha) \subseteq A \ \forall \alpha \in A$
- $\text{An}(A)$ is smallest ancestral set containing $A$ (called "ancestral hull")
  - For BN, $A$ is ancestral if $\text{an}(\alpha) \subseteq A \ \forall \alpha \in A$
  - For MRF, ancestral set is union of connectivity components, i.e., $A$ is ancestral if $\forall \alpha \in A$ we have that $\forall \beta \in [\alpha], \beta \in A$.
  - Intersection of ancestral sets is ancestral, $A, B$ ancestral then so is $A \cap B$ since if $\alpha \in (A \cap B)$ then $\text{bd}(\alpha) \subseteq A$ and $\text{bd}(\alpha) \subseteq B$, so $\text{bd}(\alpha) \subseteq (A \cap B)$.
  - Why called ancestral hull? Smallest ancestral set containing $A$:

$$\text{An}(A) = \bigcap_{B: A \subseteq B, \text{An}(B)} B$$
Ancestral examples

\( \{A, B, C, D\} \) is ancestral here... ... but not here.

- \( \text{An}(A) \neq A \cup \left( \bigcup_{\alpha \in A} \text{an}(\alpha) \right) \). Why not? We could have undirected edges (note, this is true for chain-graphs, which are like hybrid BN/MRFs).

When \( Z = \{A, B, C, D\} \),
\[
\text{An}(Z) = Z \cup \{G, F, E\} \neq Z \cup \bigcup_{\alpha \in Z} \text{an}(\alpha) = Z \cup \{F\} 
\]
Separators

- Separators are important (since they capture factorization).
- \((\alpha, \beta)\)-separator: \(C\) is an \((\alpha, \beta)\)-separator if all trails from \(\alpha\) to \(\beta\) intersect \(C\). \(C\) is a separator if it is an \((\alpha, \beta)\)-separator for some \(\alpha, \beta \in V\). \(C\) separates \(A\) from \(B\) if it is an \((\alpha, \beta)\)-separator for all \(\alpha \in A\) and \(\beta \in B\).

- minimal \((\alpha, \beta)\)-separator: if no proper subset remains an \((\alpha, \beta)\)-separator.
Tree/Forest

- **forest**: A graph $G$ is a forest if it contains no cycles (so there is a unique trail between any two vertices).
- **tree**: a tree is a connected forest.
- a *leaf* of a tree has only one adjacent edge.
- Trees are maximally cycle-free, and are minimally connected.
- trees with $n$ vertices have $n - 1$ edges.
- Trees/forests can be recognized in linear time using, e.g., DFS.
- Each tree with $\geq 2$ vertices has at least two leaves.
- Trees can be generated recursively either by attaching leaves to previously generated tree, or alternatively, by eliminating/removing leaves from previously existing tree.
- many problems are efficient on only trees (but not on other graphs, either super or subclasses of trees, such as chains, or general graphs). This includes many statistical problems (e.g., Chow-Liu tree procedure).
• *Chordal* graphs are generalizations of trees (as we will see).
• A graph $G$ is triangulated (chordal) if all cycles of length $\geq 4$ have a chord.
• which graphs are chordal?
Triangulated/Chordal

- which graphs are chordal?
Triangulated/Chordal

- which graphs are chordal?
Junction Tree

- junction tree: $\mathcal{C}$ is a collection of subsets of $V$. $\mathcal{T}$ is tree with $\mathcal{C}$ as node set, and the tree is constrained so that for any $C_1, C_2 \in \mathcal{C}$, $S = C_1 \cap C_2$ is such that $S \in C_i$ for all $C_i$ on the unique path in $\mathcal{T}$ between $C_1$ and $C_2$.

- junction tree: for any $v \in V$, the set $\{C : C \cap \{v\}\}$ induces a connected subtree in $\mathcal{T}$.

- Ex: $F$ induces a connected subtree as does any other single node.

- The intersection of any two JT nodes is on the path between the two nodes.
Junction Tree & r.i.p.

- A junction tree satisfies what is called the *running intersection property* (r.i.p.). This is often stated in two ways:
- r.i.p. The intersection of any two cliques is contained in the unique path between those two cliques in a JT.
- Subsets $C_1, C_2, \ldots$ are in r.i.p. order if the following is true.
  - $H_j = C_1 \cup C_2 \cup \ldots \cup C_j$ (history, accumulation)
  - $R_j = C_j \setminus H_{j-1}$ (innovation, residual, new stuff in $C_j$ not in previous history)
  - $S_j = H_{j-1} \cap C_j$ (non-innovation, or separator, the previous nodes that are common with the new set). Note that $C_j = R_j \cup S_j$
  - Sets are in r.i.p. order if $\forall 1 < j \leq k$, $\exists i < j : S_j \subseteq C_i$.
  - Sets are in r.i.p. order if $\forall 1 < j \leq k$, $\exists i < j : C_j \cap \{C_1 \cup \ldots \cup C_{j-1}\} \subseteq C_i$. 
Junction Tree & r.i.p.

- Sets are in r.i.p. order if $\forall 1 < j \leq k$, 
  $\exists i < j : C_j \cap \{C_1 \cup \ldots \cup C_{j-1}\} \subseteq C_i$.

- What $C_j$ has in common with all previous sets is fully contained within at least one of those previous sets.
Decomposability

- **Decomposition**: A triple \((A, B, C)\) of mutually disjoint subsets of \(V\) of \(G\) forms a decomposition of \(G\) if \(V = A \cup B \cup C\) and: 1) \(C\) separates \(A\) from \(B\), and 2) \(C\) is a clique. If \(A\) and \(B\) are non-empty, the decomposition is *proper*.
- **Decomposable**: A graph is decomposable if either: 1) it is complete, or 2) it possesses a proper decomposition \((A, B, C)\) s.t. both subgraphs \(G[A \cup C]\) and \(G[B \cup C]\) are decomposable (note this is a recursive definition).
- Ex:

  - Binary decomposition tree
Equivalent conditions for $G$

- **Thm:** The following are equivalent conditions on a graph: 
  I) $G$ is decomposable, 
  II) $G$ is chordal, 
  III) Every minimal $(\alpha, \beta)$-separator is complete.

- **Proof:** By induction on $n = |V|$. True for $n \leq 3$. Consider for $n = 4$ as exercise. Assume $n$ prove for $n + 1$.
  - **Proof:** $I \Rightarrow II$.
    - Suppose $G$ is decomposable.
    - If $G$ is complete $\Rightarrow$ chordal.
    - If $G$ not complete, $\exists$ proper decomposition $(A, B, C)$ into decomposable subgraphs $G[A \cup C]$ and $G[B \cup C]$ both with fewer vertices, $|A \cup C| < |V|$ and $|B \cup C| < |V|$.
    - $G[A \cup C]$ and $G[B \cup C]$ chordal (by inductive hypothesis).
    - Any chordless cycle (if exists) must intersect both $A$ and $B$. Since $C$ separates $A$ from $B$, chordless cycle intersects $C$ twice. But $C$ is complete, so cycle contains a chord.