Variable Elimination 2

Clique Trees

Graphical Models – 10708
Carlos Guestrin
Carnegie Mellon University
October 13th, 2006

Complexity of variable elimination – Graphs with loops

Moralize graph:
Connect parents into a clique and remove edge directions

Connect nodes that appear together in an initial factor
**Induced graph**

The induced graph $I_{F,\prec}$ for elimination order $\prec$ has an edge $X_i - X_j$ if $X_i$ and $X_j$ appear together in a factor generated by VE for elimination order $\prec$ on factors $F$.

Elimination order: 
{C,D,I,S,L,H,J,G}

**Induced graph and complexity of VE**

"Structure of induced graph encodes complexity of VE!!!"

**Theorem:**
- Every factor generated by VE subset of a maximal clique in $I_{F,\prec}$.
- For every maximal clique in $I_{F,\prec}$ corresponds to a factor generated by VE with ordering $\prec$.

**Induced width** (or treewidth)
- Size of largest clique in $I_{F,\prec}$ minus 1
- **Minimal induced width** – induced width of best order $\prec$.
**Example: Large induced-width with small number of parents**

Compact representation \(\Rightarrow\) Easy inference

Finding optimal elimination order

- **Theorem**: Finding best elimination order is NP-complete:
  - Decision problem: Given a graph, determine if there exists an elimination order that achieves induced width \(\leq K\)

- **Interpretation**:
  - Hardness of finding elimination order in addition to hardness of inference
  - Actually, can find elimination order in time exponential in size of largest clique – same complexity as inference

Elimination order: \(\{C,D,I,S,L,H,J,G\}\)
Induced graphs and chordal graphs

- **Chordal graph:**
  - Every cycle $X_1 - X_2 - \ldots - X_k - X_1$ with $k \geq 3$ has a chord
  - Edge $X_i - X_j$ for non-consecutive $i$ & $j$

- **Theorem:**
  - Every induced graph is chordal

- "Optimal" elimination order easily obtained for chordal graph

---

Chordal graphs and triangulation

- **Triangulation:** turning graph into chordal graph
- **Max Cardinality Search:**
  - Simple heuristic
  - Initialize unobserved nodes $X$ as unmarked
  - For $k = |X|$ to 1
    - $X \leftarrow$ unmarked var with most marked neighbors
    - $\prec(X) \leftarrow k$
    - Mark $X$

- **Theorem:** Obtains optimal order for chordal graphs
  - Often, not so good in other graphs!
**Minimum fill/size/weight heuristics**

- Many more effective heuristics
  - see reading
- **Min (weighted) fill heuristic**
  - Often very effective
- Initialize unobserved nodes $X$ as unmarked
- For $k = 1$ to $|X|
  - $X$ ← unmarked var whose elimination adds fewest edges
  - $\prec(X) ← k$
  - Mark $X$
  - Add fill edges introduced by eliminating $X$
- Weighted version:
  - Consider size of factor rather than number of edges

**Choosing an elimination order**

- Choosing best order is NP-complete
  - Reduction from MAX-Clique
- Many good heuristics (some with guarantees)
- Ultimately, can’t beat NP-hardness of inference
  - Even optimal order can lead to exponential variable elimination computation
- In practice
  - Variable elimination often very effective
  - Many (many many) approximate inference approaches available when variable elimination too expensive
  - Most approximate inference approaches build on ideas from variable elimination
Announcements

- Recitation on advanced topic:
  - Carlos on Context-Specific Independence
  - On Monday Oct 16, 5:30-7:00pm in Wean Hall 4615A
  - HW3 out later today

Most likely explanation (MLE)

- Query: \( \arg\max_{x_1, \ldots, x_n} P(x_1, \ldots, x_n \mid e) \)

- Using defn of conditional probs:
  \[
  \arg\max_{x_1, \ldots, x_n} P(x_1, \ldots, x_n \mid e) = \arg\max_{x_1, \ldots, x_n} \frac{P(x_1, \ldots, x_n, e)}{P(e)}
  \]

- Normalization irrelevant:
  \[
  \arg\max_{x_1, \ldots, x_n} P(x_1, \ldots, x_n \mid e) = \arg\max_{x_1, \ldots, x_n} P(x_1, \ldots, x_n, e)
  \]
Max-marginalization

\[
\max_{f,s} P(f) \cdot P(s\mid f) \cdot P(N=t\mid s)
\]

\[
\max \left[ P(F=t) \cdot P(s=t\mid F=t) \cdot P(N=t\mid S=t); \right.
\]

\[
P(F=t) \cdot P(s=t); ...
\]

\[
P(F=t) \cdot P(s=t); ...
\]

Example of variable elimination for MLE – Forward pass

\[
\max_{f,a,s,h} P(f) \cdot P(a) \cdot P(s\mid a) \cdot P(N=t\mid s)
\]

\[
= \max_{f,a,s} P(f) \cdot P(a) \cdot P(s\mid f,a) \cdot P(N=t\mid s) \cdot \max_{a} P(a)
\]

\[
= \max_{f} P(f) \cdot \max_{a} P(a) \cdot g_1(f) \cdot g_2(f,a)
\]

Interpretation:

\[
g_1(s) = \text{highest prob. achieved for each value of } s
\]

\[
g_2(f,a) = \text{highest prob. of } s \text{ and } N=t \text{ for each value of } f,a
\]

\[
g_4 = \text{prob. MLE } N=t
\]
Example of variable elimination for MLE – Backward pass

\[ F^{**} = \arg \max_f p(f) \cdot g_3(f) \]

\[ A^{**} = \arg \max_a p(a) \cdot g_2(f^{**}) \]

\[ S^{**} = \arg \max_s p(s | f^{**}, a) \cdot g_4(s) \]

\[ H^{**} = \arg \max_h p(h | s^{**}) \]

MLE Variable elimination algorithm – Forward pass

- Given a BN and a MLE query \( \max_{x_1, \ldots, x_n} P(x_1, \ldots, x_n, e) \)
- Instantiate evidence \( E = e \)
- Choose an ordering on variables, e.g., \( X_1, \ldots, X_n \)
- For \( i = 1 \) to \( n \), if \( X_i \notin E \)
  - Collect factors \( f_1, \ldots, f_k \) that include \( X_i \)
  - Generate a new factor by eliminating \( X_i \) from these factors

\[ g = \max_{x_i} \prod_{j=1}^{k} f_j \]

- Variable \( X_i \) has been eliminated!
MLE Variable elimination algorithm
– Backward pass

- \{x_1^*, \ldots, x_n^*\} will store maximizing assignment
- For \(i = n \) to 1, if \(X_i \notin \mathcal{E}\) (\(f_i\) cannot depend on \(X_i\) because \(X_{i-1}\) was eliminated)
  - Take factors \(f_1, \ldots, f_k \) used when \(X_i\) was eliminated
  - Instantiate \(f_1, \ldots, f_k\) with \(\{x_{i+1}^*, \ldots, x_n^*\}\)
    - Now each \(f_j\) depends only on \(X_i\)
  - Generate maximizing assignment for \(X_i\):

\[
x_i^* \in \arg\max_{x_i} \prod_{j=1}^{k} f_j
\]

What you need to know about VE

- Variable elimination algorithm
  - Eliminate a variable:
    - Combine factors that include this var into single factor
    - Marginalize var from new factor
  - Cliques in induced graph correspond to factors generated by algorithm
  - Efficient algorithm ("only" exponential in induced-width, not number of variables)
    - If you hear: "Exact inference only efficient in tree graphical models"
    - You say: "No!! Any graph with low induced width"
    - And then you say: "And even some with very large induced-width" (special recitation)
  - Elimination order is important!
    - NP-complete problem
    - Many good heuristics
  - Variable elimination for MLE
    - Only difference between probabilistic inference and MLE is "sum" versus "max"
What if I want to compute $P(X_i|x_0,x_{n+1})$ for each $i$?

Compute: $P(X_i| x_0, x_{n+1})$

Variable elimination for each $i$? (e.g., $X_i,\ldots,X_{i-1},X_{i+1},\ldots,X_n$

$$g_1(x_3) = \sum_{x_1} P(x_0), P(x_1|x_0), P(x_2|x_1)$$

Complexity of $P(x_i|x_0,x_{n+1}) = O(n)$

Variable elimination for every $i$, what’s the complexity?

Naive $O(n^2)$

Reusing computation

$g_2(x_5) = \sum_{x_3} P(x_3|x_2), g_1(x_3)$

$g_3(x_6) = \sum_{x_4} P(x_4|x_3), g_2(x_4)$

$g_4(x_5) = \sum_{x_4} P(x_4|x_5), P(x_4|X_5)$

Need to eliminate $X_4$: done!

Compute each message once!! (so two passes (O(n)) gives you all probs.)
Cluster graph

- **Cluster graph**: For set of factors $F$
  - Undirected graph
  - Each node $i$ associated with a cluster $C_i$
  - *Family preserving*: for each factor $f_j \in F$, $\exists$ node $i$ such that scope[$f_j$] $\subseteq C_i$
  - Each edge $i - j$ is associated with a separator $S_{ij} = C_i \cap C_j$

Factors generated by VE

- Elimination order: \{C,D,I,S,L,H,J,G\}
Cluster graph for VE

- VE generates cluster tree!
  - One clique for each factor used/generated
  - Edge $i \rightarrow j$, if $f_i$ used to generate $f_j$
  - "Message" from $i$ to $j$ generated when marginalizing a variable from $f_i$
  - Tree because factors only used once

Proposition:
- "Message" $\delta_{ij}$ from $i$ to $j$
- $\text{Scope}[\delta_{ij}] \subseteq S_{ij}$

Running intersection property

- Running intersection property (RIP)
  - Cluster tree satisfies RIP if whenever $X \in C_i$ and $X \in C_j$ then $X$ is in every cluster in the (unique) path from $C_i$ to $C_j$

Theorem:
- Cluster tree generated by VE satisfies RIP
Constructing a clique tree from VE

- Select elimination order \(≺\)
- Connect factors that would be generated if you run VE with order \(≺\)
- Simplify!
  - Eliminate factor that is subset of neighbor

Find clique tree from chordal graph

- Triangulate moralized graph to obtain chordal graph
- Find maximal cliques
  - NP-complete in general
  - Easy for chordal graphs
  - Max-cardinality search
- Maximum spanning tree finds clique tree satisfying RIP!!
  - Generate weighted graph over cliques
  - Edge weights \((i,j)\) is separator size \(-|C_i \cap C_j|\)
Clique tree & Independencies

- **Clique tree (or Junction tree)**
  - A cluster tree that satisfies the RIP
- **Theorem:**
  - Given some BN with structure $G$ and factors $F$
  - For a clique tree $T$ for $F$ consider $C_i - C_j$ with separator $S_{ij}$:
    - $X$ – any set of vars in $C_i$ side of the tree
    - $Y$ – any set of vars in $C_i$ side of the tree
  - Then, $(X \perp Y | S_{ij})$ in BN
  - Furthermore, $I(T) \subseteq I(G)$

Variable elimination in a clique tree 1

- **Clique tree for a BN**
  - Each CPT assigned to a clique
  - Initial potential $\pi_0(C_i)$ is product of CPTs
Variable elimination in a clique tree 2

- **VE in clique tree to compute** $P(X_i)$
  - Pick a root (any node containing $X_i$)
  - Send messages recursively from leaves to root
    - Multiply incoming messages with initial potential
    - Marginalize vars that are not in separator
  - Clique ready if received messages from all neighbors

Belief from message

- **Theorem**: When clique $C_i$ is ready
  - Received messages from all neighbors
  - Belief $\pi_i(C_i)$ is product of initial factor with messages:
Choice of root

- Message does not depend on root!!!

Root: node 5

```
\sum p_i(C_1) \sum p_i(C_2) \sum p_i(C_3) \sum p_i(C_4)
```

Root: node 3

```
\sum p_i(C_1) \sum p_i(C_2) \sum p_i(C_3) \sum p_i(C_4) \sum p_i(C_5)
```

“Cache” computation: Obtain belief for all roots in linear time!!

Shafer-Shenoy Algorithm
(a.k.a. VE in clique tree for all roots)

- Clique $C_i$ ready to transmit to neighbor $C_j$ if received messages from all neighbors but $j$
  - Leaves are always ready to transmit
- While $\exists C_i$ ready to transmit to $C_j$
  - Send message $\delta_{i \rightarrow j}$
- Complexity: Linear in # cliques
  - One message sent each direction in each edge
- Corollary: At convergence
  - Every clique has correct belief
Calibrated Clique tree

- Initially, neighboring nodes don’t agree on “distribution” over separators
- **Calibrated clique tree:**
  - At convergence, tree is *calibrated*
  - Neighboring nodes agree on distribution over separator

Answering queries with clique trees

- **Query within clique**
  - Incremental updates – Observing evidence \( Z=z \)
    - Multiply some clique by indicator \( \mathbf{1}(Z=z) \)

- **Query outside clique**
  - Use variable elimination!
Message passing with division

- Computing messages by multiplication:

- Computing messages by division:

Lauritzen-Spiegelhalter Algorithm (a.k.a. belief propagation) Simplified description see reading for details

- Initialize all separator potentials to 1
  - $\mu_{ij} \leftarrow 1$

- All messages ready to transmit

- While $\exists \delta_{i \rightarrow j}$ ready to transmit
  - $\mu_{ij}' \leftarrow$
  - If $\mu_{ij}' \neq \mu_{ij}$
    - $\delta_{i \rightarrow j} \leftarrow$
    - $\pi_j \leftarrow \pi_j \times \delta_{i \rightarrow j}$
    - $\mu_j \leftarrow \mu_j'$
    - $\forall$ neighbors $k$ of $j$, $k \neq i$, $\delta_{j \rightarrow k}$ ready to transmit

- Complexity: Linear in # cliques
  - for the "right" schedule over edges (leaves to root, then root to leaves)

- Corollary: At convergence, every clique has correct belief
VE versus BP in clique trees

- VE messages (the one that multiplies)
- BP messages (the one that divides)

Clique tree invariant

- **Clique tree potential**: Product of clique potentials divided by separators potentials
- **Clique tree invariant**: $P(X) = \pi_T(X)$
Belief propagation and clique tree invariant

- **Theorem**: Invariant is maintained by BP algorithm!

- BP reparameterizes clique potentials and separator potentials
  - At convergence, potentials and messages are marginal distributions

Subtree correctness

- **Informed message** from i to j, if all messages into i (other than from j) are informed
  - Recursive definition (leaves always send informed messages)

- **Informed subtree**:
  - All incoming messages informed

- **Theorem**:
  - Potential of connected informed subtree $T'$ is marginal over $\text{scope}[T']$

- **Corollary**:
  - At convergence, clique tree is calibrated
    - $\pi_i = P(\text{scope}[\pi_i])$
    - $\mu_{ij} = P(\text{scope}[\mu_{ij}])$
Clique trees versus VE

Clique tree advantages
- Multi-query settings
- Incremental updates
- Pre-computation makes complexity explicit

Clique tree disadvantages
- Space requirements – no factors are “deleted”
- Slower for single query
- Local structure in factors may be lost when they are multiplied together into initial clique potential

Clique tree summary
- Solve marginal queries for all variables in only twice the cost of query for one variable
- Cliques correspond to maximal cliques in induced graph
- Two message passing approaches
  - VE (the one that multiplies messages)
  - BP (the one that divides by old message)
- Clique tree invariant
  - Clique tree potential is always the same
  - We are only reparameterizing clique potentials
- Constructing clique tree for a BN
  - from elimination order
  - from triangulated (chordal) graph
- Running time (only) exponential in size of largest clique
  - Solve exactly problems with thousands (or millions, or more) of variables, and cliques with tens of nodes (or less)