Our two main texts


Announcements

- Reading assignment: Read the tree_inference.pdf chapter soon to be posted on the web page.
- Read chapters 1, 5, and 6 in the MRF book.
- Read chapter 1 in the Wainwright and Jordan book.
- At https://catalyst.uw.edu/gopost/board/bilmes/23863/ you can find a draft copy of tree_inference.pdf and evidence.pdf

Class Road Map

We need to find one makeup lecture this term.

- L1 (9/28): Introduction, Families, Semantics
- LX (9/30): No class
- L2 (10/5): Trees, exact inference
- L3 (10/7): More on trees and inference.
- L4 (10/12): To tree or not to tree.
- L5 (10/14): All models lead to trees
- L6 (10/19):
- L7 (10/21):
- L8 (10/26):
- L9 (10/28):
- L10 (11/2):
- L11 (11/4):
- L12 (11/9):
- LXX (11/11): Veterans Day, no class
- L13 (11/16):
- L14 (11/18):
- L15 (11/23):
- LXX (11/25): Thanksgiving, no class
- L16 (11/30):
- L17 (12/2):
- L18 (12/7):
- L19: (12/9):
Non-tree graphs: effectively doing inference on perfect elimination graph.

After elimination, we've got a perfect (fill-in free) elimination graph.

We encounter the maxcliques when we run elimination.

Elimination cliques are superset of set of maxcliques.

We may embed any $G = (V, E)$ into any $G = (V, E \cup F)$.

Given perfect elimination graph, easy to find perfect elimination order.

Triangulated graphs (chordal), all cycles are chorded.

Various definitions of separators.

Triangulated iff all min separators are complete.

Any triangulated graph on $\geq 2$ nodes has two simplicial nodes.

In a triangulated graphs, all nodes simplicial?

If $G$ is triangulated and $v$ simplicial, if we eliminate $v$, is $G[V \setminus v]$ still triangulated?

Therefore:

**Corollary 1**

*For any triangulated graph, there exists an elimination order that does not produce any fill in.*

So if we know the graph is triangulated, we can easily find a perfect elimination order. Why? We can strengthen the above in fact:
Lemma 2

If $G$ is a graph and there exists a perfect elimination order, then $G$ is triangulated.

Proof.

By induction. It is obviously true for 1 and 2 nodes. Assume true for $n$ nodes, and we are given an $n + 1$ node graph. Since there exists an elimination order without fill-in, there exists a simplicial node, where chorded cycles can not exist through that node since all of its neighbors are connected. Once we eliminate that node, no fill-in is created, and induction step applies.

We summarize the bijection as follows:

Theorem 3

A graph $G$ is triangulated iff there exists a perfect elimination order over the nodes in $G$.

Corollary 4

Take any graph $G$ and an elimination order $\sigma$, then the reconstituted graph $G' = (V, E \cup F_\sigma)$ is triangulated.
Therefore, we can generate a reconstituted elimination graph (or any triangulated graph) using a reverse elimination order.

**Algorithm 1:** Regenerate triangulated graph.

**Input:** A triangulated graph $G = (V, E)$ and a perfect elimination order $\sigma$  
**Result:** A new graph $G'$ identical to $G$.

1. Recall that $\delta_{G_{i-1}}(\sigma_i)$ are neighbors of $G$ at the point $\sigma_i$ is eliminated.
2. Add $\sigma_N$ to $V(G')$;
3. for $i = N - 1 \ldots 1$ do
   4. Add $\sigma_i$ to $V(G')$;
   5. Add $\delta_{G_{i-1}}(\sigma_i)$ to $E(G')$;  /* at this $\delta_{G_{i-1}}(\sigma_i)$ is complete */

**Theorem 5**

Let $G = (V, E)$ be a graph and let $G' = (V, E \cup F)$ be a triangulation of $G$ with $F$ the required edge fill-in. If the triangulated graph is minimal in the sense that for any $F' \subset F$, the graph $G'' = (V, E \cup F')$ is no longer triangulated, then $F$ can be obtained by the result of an elimination order. That is, the elimination algorithm and the various variable orderings may produce all minimal triangulations of a graph $G$.

- Minimal triangulations are state-space optimal for positive distributions only!
Triangulated vs. Perfect elimination graphs

Minimal triangulations are state-space optimal for positive distributions only. Let $d$ be a deterministic function of $a$ and $b$. All variables have $r$ values but $d$ has $r^2 - 1$ values.

Moralized already chordal, perfect elim. order $(c, a, b, d)$. One clique at $O(r^2)$, two at $O(r^4)$.

Elimination order $(a, c, b, d)$, cost is still $O(r^4)$

Start by eliminating $d$, cost is still $O(r^4)$

Triangulation unobtainable with elimination, cost $O(r^3)$.

re-cap

- We wish to run elimination
- Doing so produces a triangulated graph
- Complexity is its largest clique in result
- We encounter the cliques (and the largest) during elimination so we get the complexity while we are doing elimination
- Elimination adds edges, we can embed original graph into resulting triangulated graph (triangulated graph covers original graph)
- can’t avoid a triangulated graph — always dealing with triangulated graphs implicitly or explicitly
- want to find covering minimally triangulated graph with smallest largest maxclique
- i.e., find optimal elimination order
- there are $n!$ elimination orders
- is this easy or hard? We shall see . . .
Generalizations of a tree as defined as follows:

**Definition 6 (k-tree)**

A complete graph with $k + 1$ nodes is a $k$-tree. To construct a $k$ tree with $n + 1$ nodes starting from a $k$-tree with $n$ nodes, choose some size $k$ complete sub-graph of the $n$-node $k$-tree and connect the $n + 1$'st node to all nodes in the $k$-node complete sub-graph.

- Any complete $n$-graph is an $n − 1$-tree
- a regular tree is a 1-tree.
- all $k$-trees are triangulated
### k-trees

- In a tree, all minimal separators are size 1.
- In a $k$-tree, all minimal separators are size $k$ (and thus a $k$-clique).
- In a $k$-tree, all maxcliques are size $k + 1$, so maximum clique size is $k + 1$.
- In a $k$-tree, complexity of inference will be $O(r^{k+1})$.

#### even stronger:

#### Lemma 7

A graph $G = (V, E)$ is a $k$-tree iff

- $G$ is connected
- $G$’s maximum clique is of size $k + 1$
- Every minimal separator of $G$ is a $k$-clique.
**k-trees**

**Definition 8 (partial k-tree)**

Any spanning sub-graph of a k-tree is a partial k-tree.

- Any partial k-tree is embeddable into a k-tree.
- Inference in a partial k-tree is at most $O(r^{k+1})$.
- k-trees are triangulated, but arbitrary triangulated graph not necessarily a k-tree
- any triangulated graph can be embedded into a k tree for large enough k — silly example, set $k = (n - 1)$.
- But is it possible for smaller k?

**k-trees and embeddings**

**Lemma 9**

*If G is a triangulated graph with at least $k + 1$ vertices and has a maximum clique of size at most $k + 1$, then G can be embedded into a k-tree.*

**Proof.**

Let $\sigma = (\sigma_1, \ldots, \sigma_n)$ be perfect elim. order for $G$. We embed $G$ into k-tree by adding edges to $G$ so that same ordering is perfect in the k-tree.

Induction.

Base case, any set of $k + 1$ vertices can be embedded into a k tree by making those vertices a clique. Thus, add edges to last $k + 1$ eliminated vertices, i.e., make $\{\sigma_{n-k}, \sigma_{n-k+1}, \ldots, v_n\}$ a $k + 1$-clique. ...
... proof continued.

Induction: assume the subgraph with vertices \( \{\sigma_{i+1}, \ldots, \sigma_n\} \) has been embedded into a \( k \)-tree \( T_{i+1} \). Since the maximum clique size of \( G \) is \( k + 1 \), in \( G \) vertex \( \sigma_i \) is adjacent to a clique \( c \) with no more than \( k \) vertices in \( \{\sigma_{i+1}, \ldots, v_n\} \). In the \( k \)-tree \( T_{i+1} \), \( c \) is contained in a \( k \)-clique \( c' \). When we make \( v_i \) adjacent to all of the vertices of \( c' \), we obtain a \( k \)-tree \( T_i \) since \( v_i \) is still simplicial in \( T_i \). Repeating to \( v_1 \) and result is supergraph of \( G \) with same order being perfect.

\[ \]

\( k \)-trees and embeddings

- Therefore, reconstituted elimination graph can be embedded into a \( k \)-tree for large enough \( k \).
- Note: in \( k \)-tree all cliques are size \( k + 1 \) but in our reconstituted perfect elimination graph, might only have one clique that is of size \( k + 1 \) (but even given this, we will still show a negative result on the next slide 😊)
- Goal: We want to find the elimination order that results in the smallest \( k \) such that \( G' = (V, E \cup F_\sigma) \) can be embedded into a \( k \)-tree.
- i.e., find best “Chordal cover”
- Unfortunately:
**Goal:** We want to find the elimination order that results in the smallest $k$ such that $G' = (V, E \cup F_\sigma)$ can be embedded into a $k$-tree, i.e., find best “Chordal cover”

**Theorem 10**

*For an arbitrary graph $G = (V, E)$, finding the smallest $k$ such that $G$ can be embedded into a $k$-tree is an NP-complete optimization problem (i.e., the decision version of the problem, asking if $G$ can be embedded into a $k$-tree of size $k$, is NP-complete).*

- consider again elimination as summing out variables - not possible to guarantee optimal summation in poly-time order unless $P=NP$.
- We resort to heuristics (min fill, min size, random chose from top $\ell$ with random restarts, etc. work well).
- Inapproximability result: (see below)

**Heuristics for elimination**

Since we can’t expect to find a perfect elimination order, we have heuristics:

- **min fill-in heuristic:** Eliminate next the node $n$ that would result in the smallest number of fill-in edges at that step. Break ties arbitrarily.
- **min size heuristic:** Eliminate next the node that would result in the smallest clique when eliminated (i.e., choose the node as one with the smallest edge degree). Break ties arbitrarily.
- **min weight heuristic:** If the nodes have non-uniform domain sizes, then we choose next the node that would result in the clique with the smallest state space, which is defined as the product of the domain sizes. Break ties arbitrarily.
Better Heuristics for elimination

Variants and improvements to the above heuristics.

- **tie-breaking**: When one heuristic has tie, choose one of the other heuristics to break tie.

- **non-greedy**: Rather than greedily choosing best vertex, take the $m$-best vertices (e.g., the $m < n$ nodes that would result in, say, the smallest fill-in) and eliminate one of them.

- **random next step**: Create a distribution over those $m$-best vertices, where the probability formed by either: 1) uniform, or 2) inversely proportional to the greedy score (e.g., inverse fill-in). Draw from this distribution to choose node to eliminate.

- **random repeats**: Run above heuristics multiple times, producing different elimination orders. Choose one that results in the smallest maximum clique size.

$k$-trees and embeddings

- Goal: We want to find the elimination order that results in the smallest $k$ such that $G' = (V, E \cup F_\sigma)$ can be embedded into a $k$-tree, i.e., find best “Chordal cover”

**Theorem 10**

For an arbitrary graph $G = (V, E)$, finding the smallest $k$ such that $G$ can be embedded into a $k$-tree is an NP-complete optimization problem (i.e., the decision version of the problem, asking if $G$ can be embedded into a $k$-tree of size $k$, is NP-complete).

- consider again elimination as summing out variables - not possible to guarantee optimal summation in poly-time order unless P=NP.

- We resort to heuristics (min fill, min size, random chose from top $\ell$ with random restarts, etc. work well).

- Inapproximability result: (see below)
Other views of the difficulty

There are a class of related problems that equivalently indicate the difficulty were are in.

**Theorem 11**

*Given an arbitrary graph $G = (V, E)$, find the largest clique $C \subseteq V(G)$, where large is measured in terms of $|C|$ is an NP-complete optimization problem.*

- Approximation algorithms - possible to do no worse than $O((\log |V|)^2/|V|)$ times size of true maximum size clique.
- Inapproximable $|V|^{1/2-\epsilon} \text{ for any } \epsilon > 0.$
- If we could find the smallest $k$ such that it could be embedded it a $k$ tree, we could identify the maximum clique in the graph. How?

Another view of the difficulty

While we’re at it, even finding best chordal fill-in is hard

**Theorem 12**

*Given an arbitrary graph $G = (V, E)$, and $G' = (V, E \cup F)$ is a triangulation of $G$, finding the smallest such $F$ is an NP-complete optimization problem.*

Thus, to summarize, finding the optimal elimination order is likely computationally hard, as are other problems associated with graphs.
Some good news 😊 - at least we can identify triangulated graphs

- We know that if there is a perfect elim order, the graph is triangulated.
- keep eliminating simplicial nodes as long as you can, and output “not triangulated” if ever there is no simplicial node.
- naïve implementation: find fill in of each node, eliminate the one with no fill-in $O(n^3)$.
- There is a smart algorithm, maximum cardinality search (MCS), that can do this in $O(|V| + |E|)$
- Basic idea of MCS: produce a perfect elimination order, if it exists, in reverse. Construct it by looking at previously labeled neighbors.

MCS

**Input:** An undirected graph $G = (V, E)$ with $n = |V|$.
**Result:** triangulated or not, MCS ordering $\sigma = (v_1, \ldots, v_n)$

```
1 L ← ∅ ; i ← 1 ;
2 while |V \ L| > 0 do
3     Choose $v_i \in \arg\max_{u \in V \setminus L} |\delta(u) \cap L|$ ; /* $v_i$’s previously labeled neighbors has max cardinality. */
4     $c_i ← \delta(v_i) \cap L$ ; /* $c_i$ is $v_i$’s neighbors in the reverse elimination order. */
5     if $\{v_i\} \cup c_i$ is not complete in $G$ then
6         return “not triangulated” ;
7     L ← L ∪ $\{v_i\}$ i ← i + 1 ;
8 return “triangulated”, the node ordering $\sigma$
```
Can also produce an elimination order and triangulate the graphs (but not particularly good)
will produce a perfect elimination order on triangulated graphs
why called maximum cardinality “search”

**Theorem 13**

A graphical $G$ is triangulated iff in the MCS algorithm, at each point when a vertex is marked, that vertex’s previously marked neighbors form a complete subgraph of $G$.

**Corollary 14**

Every maximum cardinality search of a triangulated graph $G$ corresponds to a reverse perfect eliminating order of $G$.

Multiple queries

Like in tree case, we may be interested in more than just $p(x_L)$ and using elimination for $p(x_{V\setminus L})$.
Ideally, we would like to use all this work in producing elimination orders for more than just one query.

Tree inference case - messages for one query could be used for other queries. Amount of message re-use only grows with number of queries.
Goal: do the same thing here.
elevation to class of triangulated models is unavoidable, even for one query, so at this point we need not consider anything outside the class of triangulated models.
But is one triangulated model optimal for all queries?
Multiple queries

- A triangulated graph is a cover of \( G \)
- Any clique in \( G \) will still be a clique in a triangulation \( G' \)
- Given clique \( c \in \mathcal{C}(G) \), there exists \( c' \in \mathcal{C}(G') \) with \( c \subseteq c' \).
- Given \( p(x_{c'}) \), can compute \( p(x_c) = \sum_{x_{c'} \setminus c} p(x_c) \) at \( O(r|c'|) \), same cost triangulated graph.
- Optimal \( k \)-tree embedding for \( G \) is one that minimizes the maximum clique for any triangulation of \( G \), so if we have found this embedding, this will be optimal for any original-graph clique marginal.
- Even if we found a “good” elimination order (one that produces a maxclique of reasonable size), this order can be shared for other clique queries.

Non-clique queries

- Recall: 1-tree case, if we want a marginal over a non-sub-tree, we might be in trouble.
- Similarly, if we desire non-clique queries for general graph, then computation can get worse. Computing \( p(x_L) \) for arbitrary \( L \) could turn \( x_L \) into a clique in the worst case (Rose’s theorem).
- If \( x_L \) is not clique in \( G' \), then we can view \( G' \) as not being “valid” for the query \( p(x_L) \).
- In such case, need to re-triangulate, starting with a graph where \( x_L \) is made complete.
clique queries

- Remarkably, in the case of clique queries, we can actually re-use the elimination order.
- But like in tree-case, we want to share more than just the elimination order.
- Goal: in non-tree graphs, re-use work of actually computing marginals for multiple queries.
- We’ll see an amazing fact: if we find the optimal elimination order for 1 clique query, it is optimal for all clique queries!! 😊

Decomposition of $G$

**Definition 15 (Decomposition of $G$)**

A *decomposition* of a graph $G = (V, E)$ (if it exists) is a partition $(A, B, C)$ of $V$ such that:

- $C$ separates $A$ from $B$ in $G$.
- $C$ is a clique.

If $A$ and $B$ are both non-empty, then the decomposition is called *proper*.

If $G$ has a decomposition, what does this mean for the family $\mathcal{F}(G, R^{(f)})$? Since $C$ separates $A$ from $B$, this means that $X_A \perp X_B | X_C$ for any $p \in \mathcal{F}(G, R^{(f)})$, which moreover means we can write the joint distribution in a particular form.

$$p(x_A, x_B, x_C) = \frac{p(x_A, x_C)p(x_B, x_C)}{p(x_C)} \quad (1)$$
Definition 16

A graph $G = (V, E)$ is decomposable if either: 1) $G$ is a clique, or 2) $G$ possesses a proper decomposition $(A, B, C)$ s.t. both subgraphs $G[A \cup C]$ and $G[B \cup C]$ are decomposable.

- Note that the separator is contained within the subgraphs: i.e., $G[A \cup C]$ rather than, say, $G[A]$. 

---

Graph and two decompositions of this graph.
Decomposable models

- Internal nodes in tree are complete graphs that are also separators.
- With $G$ is decomposable, what are implications for a $p \in \mathcal{F}(G, R^{(f)})$?

\[
p(A, B, C, D, E, F, G, H, I, J, K) \\
= \frac{p(A, C, D, F) p(B, C, D, E, F, G, H, I, J, K)}{p(C, D, F)} \\
= \frac{p(A, C, D, F)}{p(C, D, F)} \left( \frac{p(B, C, G, H) p(C, D, E, F, H, I, J, K)}{p(C, H)} \right) \\
= \ldots \\
= \frac{p(A, C, D, F) p(B, G, H) p(C, B, H) p(I, E, J) p(E, I, D) p(C, K, H) p(D, K, I) p(D, K, F, C)}{p(C, D, F) p(C, H) p(B, H) p(D, I) p(E, I) p(C, K) p(D, K)}
\]

- $S$ is a separator, so that $G[V \setminus S]$ consists of 2 or more connected components.
- We say that $S$ shatters the graph $G$ into those components, and let $d(S)$ be the number of connected components that $S$ shatters $G$ into. $d(S)$ is the shattering coefficient of $G$.
- Example: below, $d(\{A, B\}) = 3$
Decomposable models

- When $d(S) > 2$, separator marginal use more than once in the denominator
- The general form of the factorization becomes:

$$p(x) = \frac{\prod_{C \in C(G)} p(x_C)}{\prod_{S \in S(G)} p(x_S)^{d(S)-1}}$$  \hspace{1cm} (2)

- Any decomposable model can be written this way
- 4-cycle is not decomposable. Two independence properties that can’t be used simultaneously.

$$p(x_1, x_2, x_3, x_4) = \frac{p(x_1, x_2, x_4)p(x_1, x_3, x_4)}{p(x_1, x_4)} = \frac{p(x_1, x_2, x_3)p(x_2, x_3, x_4)}{p(x_2, x_3)}$$  \hspace{1cm} (3)

Proposition 17

*All of the maxcliques in a graph lie on the leaf nodes of the binary decomposition tree*

Proof.

For a decomposable model, the base case (leaf node) is a clique, otherwise it would not be decomposable. If a leaf was not a maxclique, then that means it is contained in a maxclique, and got split by a separator corresponding to that leaf’s parent, but this is impossible since a maxcliques have no separator.

Proposition 18

*The set of all minimal separators of graph constitute the unique non-leaf nodes of the binary decomposition tree, with $d(S) - 1$ being the number of times the minimal separator appears as a given non-leaf node.*
A bit of notation

- If \( C \) is separator, \( C \) shatters \( G \) into \( d(C) \) connected components
- \( G[V \setminus C] \) is the union of these components
- Let \( \{G_1, G_2, \ldots, G_\ell\} \) be (disjoint) connected components of \( G[V \setminus C] \), so
  \( G_1 \cup G_2 \cup \cdots \cup G_\ell = G[V \setminus C] \)
- Given \( a \in V(G_i) \) for some \( i \), then \( G[V \setminus C](a) = G_i \).

Triangulated vs. decomposable

**Theorem 19**

A given graph \( G = (V, E) \) is triangulated iff it is decomposable.

**Proof.**

First: decomposability implies triangulated (induction). Next: every minimal separator complete in \( G \) implies decomposable.

Assume \( G \) is decomposable. If \( G \) is complete then it is triangulated. If it is not complete then there exists a proper decomposition \( (A, B, C) \) into decomposable subgraphs \( G[A \cup C] \) and \( G[B \cup C] \) both of which have fewer vertices, meaning \( |A \cup C| < |V| \) and \( |B \cup C| < |V| \). By the induction hypothesis, both \( G[A \cup C] \) and \( G[B \cup C] \) are chordal. Any potential chordless cycle, therefore, can’t be contained in one of the sub-components, so if it exist in \( G \) must intersect both \( A \) and \( B \). Since \( C \) separates \( A \) from \( B \), the purported chordless cycle would intersect \( C \) twice, but \( C \) is complete the cycle has a chord. ...
Triangulated vs. decomposable

... proof continued.

Next, assume that all minimum \((a, b)\) separators are complete in \(G\). If \(G\) is complete then it is decomposable. Otherwise, there exists two non-adjacent vertices \(a, b \in V\) in \(G\) with a necessarily complete minimal separator \(C\) forming a partition \(G[V \setminus C](a), G[V \setminus C](b)\), and all of the remaining components of \(G[V \setminus C]\). We merge the connected components together to form only two components as follows: let \(A = G[V \setminus C](a) \cup D\) and \(B = G[V \setminus C](b)\). Since \(C\) is complete, we see that \((A, B, C)\) form a decomposition of \(G\), but we still need that \(G[A \cup C]\) and \(G[B \cup C]\) to be decomposable (see figure).
... proof continued.

Let $C_1$ be a minimal $(a_1, b_1)$ separator in $G[A \cup C]$. But then $C_1$ is also a minimal $(a_1, b_1)$ separator in $G$ since, once we add $B$ back to $G[A \cup C]$ to regenerate $G$, there still cannot be any new paths from $a_1$ to $b_1$ circumventing $C_1$. This is because any such path would involve nodes in $B$ (the only new nodes) which, to reach $B$ and return, requires going through $C$ (which is complete) twice. Such a path cannot bypass $C_1$ since if it did, a shorter path not involving $B$ would bypass $C_1$. Therefore, $C_1$ is complete in $G$, and an inductive argument says that $G[A \cup C]$ is decomposable. The same argument holds for $G[B \cup C]$. Therefore, $G$ is decomposable.

---

**Tree decomposition**

**Definition 20 (tree decomposition)**

Given a graph $G = (V, E)$, a tree-decomposition of a graph is a pair \( \{C_i : i \in I\}, T \) where $T = (I, F)$ is a tree with node index set $I$, edge set $F$, and $\{C_i\}_i$ (one for each $i \in I$) is a collection of subsets of $V(G)$ such that:

1. $\bigcup_{i \in I} C_i = V$
2. for any $(u, v) \in E(G)$, there exists $i \in I$ with $u, v \in C_i$
3. for any $v \in V$, the set $\{i \in I : v \in C_i\}$ forms a connected subtree of $T$
Tree decomposition is also hard

- The tree-width of the tree-decomposition is the size of the largest $C_i$ minus one (i.e., $\max_{i \in I} |C_i| - 1$).

**Theorem 21**

*Given graph $G = (V, E)$, finding the tree decomposition $T = (I, F)$ of $G$ that minimizes the tree width ($\max_{i \in I} |C_i| - 1$) is an NP-complete optimization problem.*

- Note that this is approximable within $O(\log |V|)$ but it is not possible to better than $|V|^{1-\epsilon}$ for any $\epsilon > 0$.
- How does this relate to our problem though?
Most of this material comes from the tree_inference.pdf handout posted at https://catalyst.uw.edu/gopost/board/bilmes/23863/.