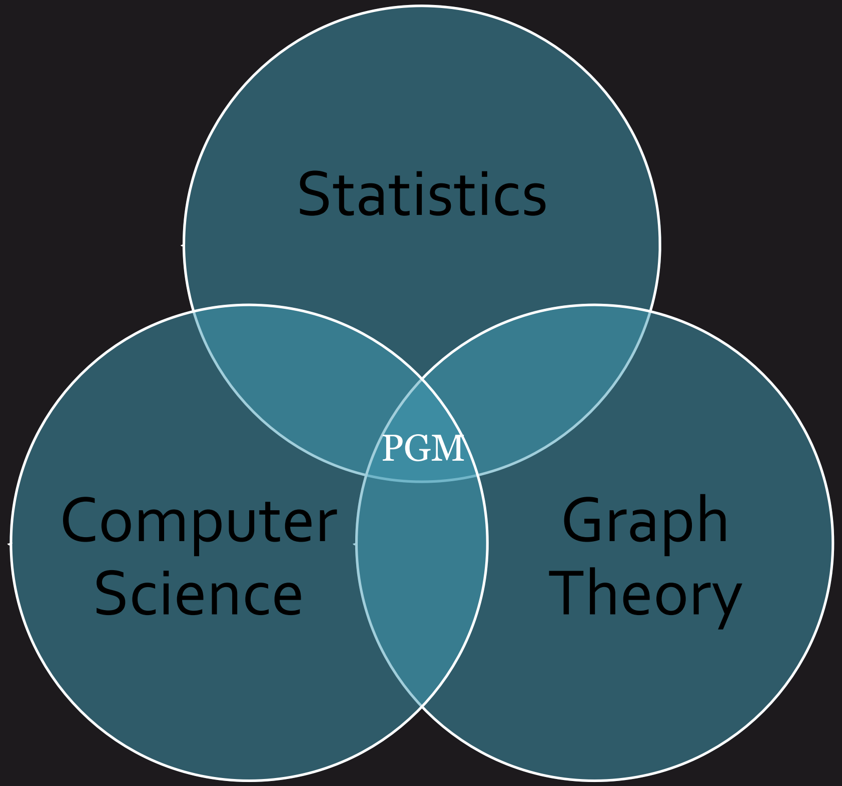
Structure Learning Algorithms for Chain Graphs

Damian Skrzypiec

19.12.2017

Probabilistic Graphical Models

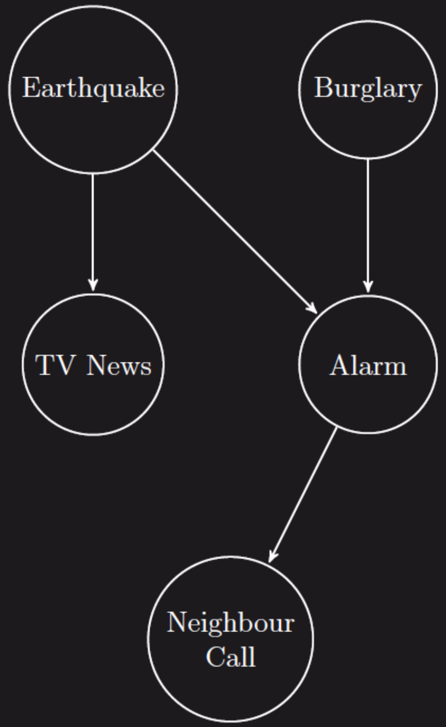
A **probablilistic graphical model** (Graphical Model) is a probabilistic model for which a graph expresses the conditional dependence structure between random variables.



Bayesian Networks

Bayesian network is a probabilistic graphical model which is represented by a directed acyclic graph (DAG). It is the most common class of PGMs. Bayesian networks were formally introduced in 1982 by Pearl.

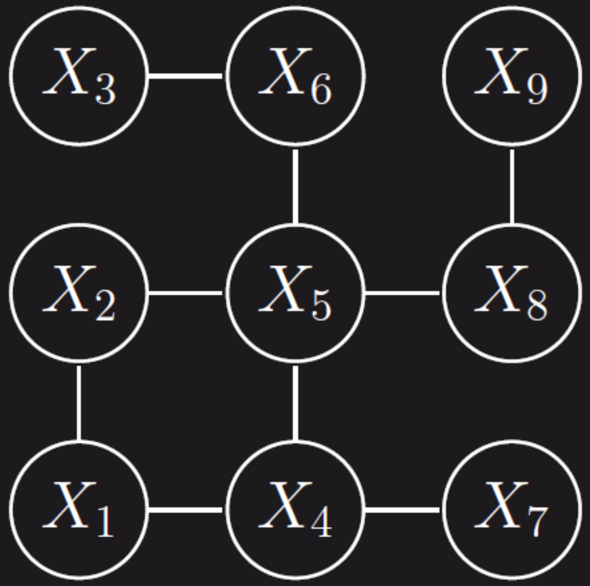
$$p(x) = \prod_{v \in V} p\left(x_v \mid x_{pa(v)}\right)$$



Markov Random Fields

Markov Random Field is a probabilistic graphical model which is represented by an undirected graph. Besides bayesian networks it is one of the most common class of PGMs.

$$p(x) = \prod_{c \in Cliques} F_c(x)$$



Why PGMs?

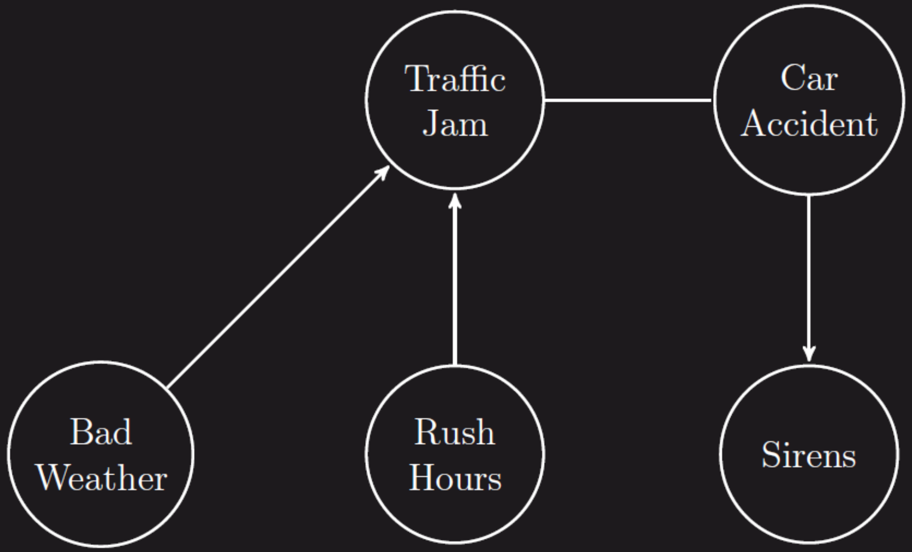
- Graphical representation is very intuitive and convenient in case of high dimension probability spaces

- Graphical representation provides insight about conditional independence structure just by looking at the graph.

- Graphical representation makes easier to requires less comuputational to calculate conditional probability (e.g **P**(Earthquake | Alarm = 1))

Chain Graphs

Chain graphs is a class of PGMs which is represented by graphs not containing cycles. It can contains both directed and undirected edges. Thus chain graphs can be perceived as generalization of bayesian networks and Markov fields.



Parametrization of chain graphs

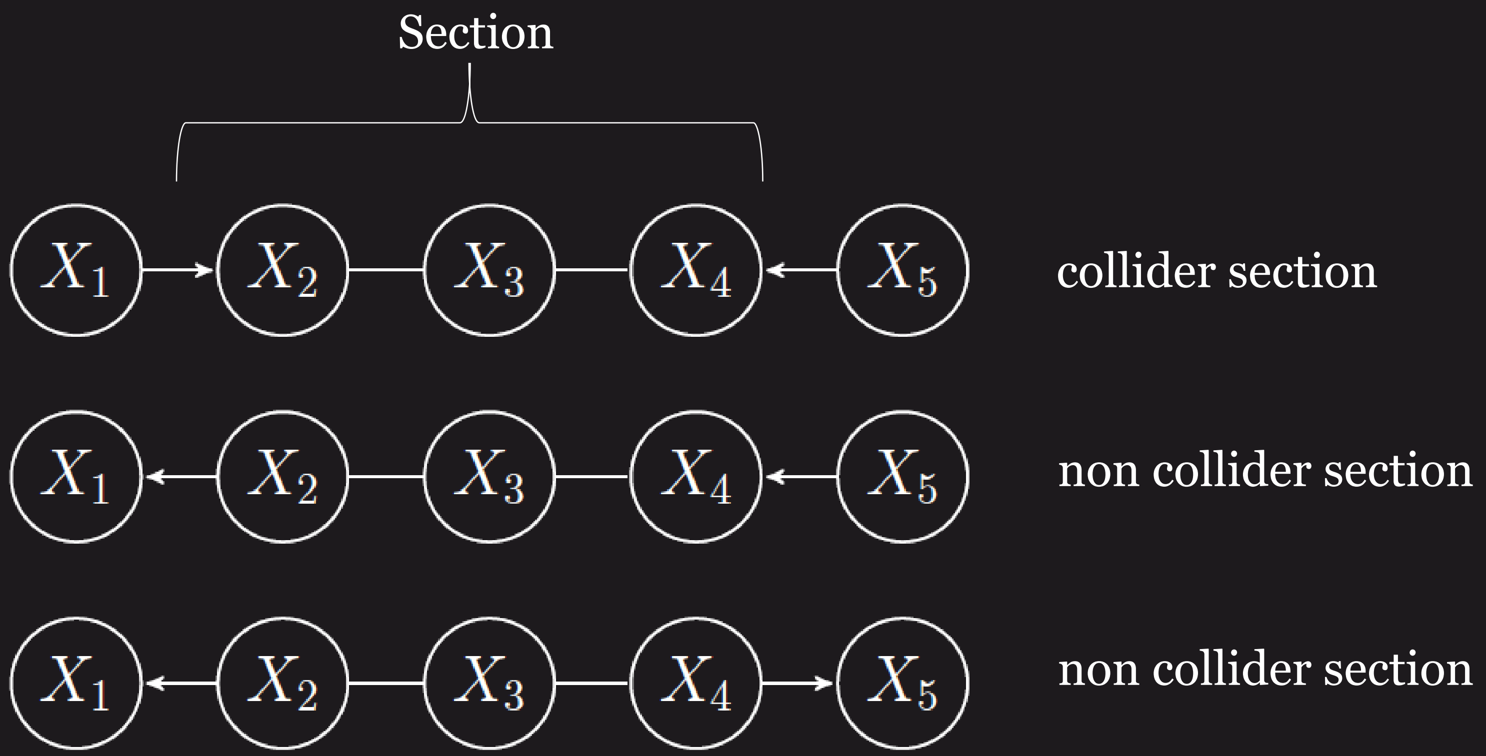
A **chain component C** of chain graph is a maximal set of vertices such that there is a path between every pair of vertices in **C** containing only undirected edges. When we treat chain components of chain graph as „nodes" we got DAG. Therefore

For chain components $C_1, C_2, \ldots, C_k$ of chain graph $G = (V, E)$

$$p(x) = \prod_{i \in \{1,2,\ldots,k\}} p\left(C_i \mid pa(C_i)\right)$$

$$p\left(C_i \mid pa(C_i)\right) = \frac{1}{\mathrm{Const}} \prod_{M \in Clique(C_i)} F_M(x_M)$$

C-Separation



C-Separation

Definition. *(Intervention)*
A route ρ in graph $G = (V, E)$ is blocked by a subset $S \subset V$ of vertices if and only if there exists a section σ of route ρ such that one of the following conditions is satisfied.

1. *Section σ is a collider section with respect to ρ and σ is outside of S.*
2. *Section σ is non collider section with respect to ρ and σ is hit by S.*

C-Separation

Definition. *(c-separation)*
Let $G = (V, E)$ *be a chain graph. Let* A, B, S *be three disjoint subsets of the vertex set* V*, such that* A *and* B *are nonempty. We say that* A *and* B *are c-separated by* S *on* G *if every route within one of its terminals in* A *and the other in* B *is blocked by* S*. We call* S *a c-separator for* A *and* B *and mark as* $\langle A, B \mid S \rangle_{\mathcal{G}}^{sep}$.

Separation Trees

Definition. *(Node Tree)*
Let $G = (V, E)$ *be a graph and* $\mathcal{C} = \{C_1, \ldots, C_k\}$ *be a node set of graph* G. *A node tree is a graph* $\mathcal{T}(G, \mathcal{C}) = (\mathcal{C} \cup \mathcal{S}, E)$, *where* $\mathcal{S} = \{C_i \cap C_j \mid i, j \in \{1, 2, \ldots, k\}\}$ *is set of so-called separators and* $E = \{C_i - C_j \mid C_i \cap C_j \neq \emptyset$ *and* $i, j \in \{1, 2, \ldots, k\}\}$ *is set of undirected edges.*

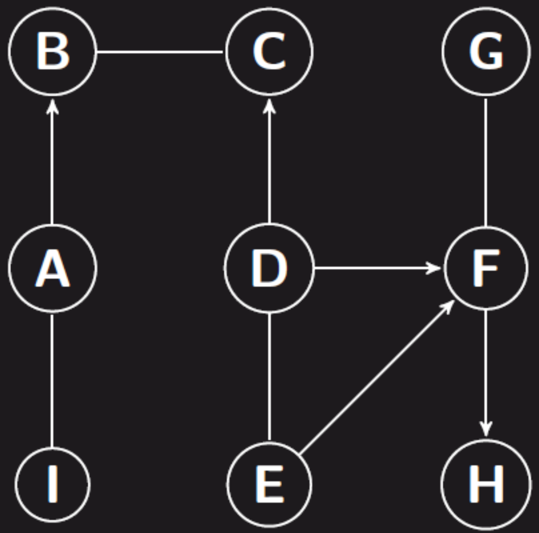
Definition. *(Separation tree)*
For given chain graph $G = (V, E)$ *and node set* $\mathcal{C}$ *we say that node tree* $\mathcal{T}(G, \mathcal{C})$ *is a separation tree if*

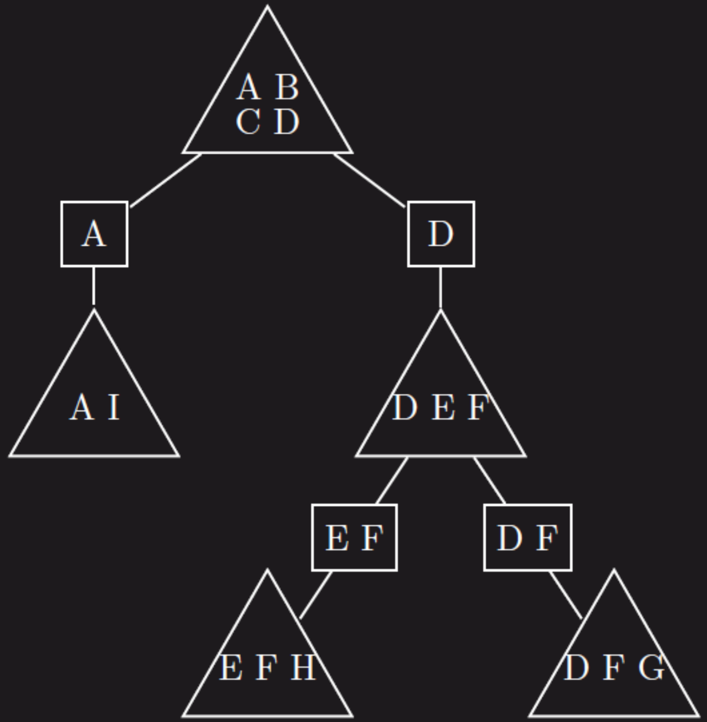
1. $\bigcup_{C \in \mathcal{C}} C = V$ *and*
2. *for any separator* S *in node tree* $\mathcal{T}(G, \mathcal{C})$ *we have*

$$\langle V_1(S) \setminus S, V_2(S) \setminus S \mid S \rangle_{\mathcal{G}}^{sep}$$

Separation Trees

$$\mathcal{C} = \{\{A, I\}, \{A, B, C, D\}, \{D, E, F\}, \{D, F, G\}, \{E, F, H\}\}.$$





Main Theorem

Theorem. *Let $\mathcal{T}(G, \mathcal{C})$ be a separation tree for chain graph G. Then vertices u and v are c-separated by some set $S_{uv} \subset V$ in G $(*)$ if and only if one the following conditions hold:*

1. *Vertices u and v are not contained together in any node C of $\mathcal{T}(G, \mathcal{C})$,*
2. *Vertices u and v are contained together in some node C, but for any separator S connected to C, $\{u, v\} \not\subset S$, and there exists $S'_{uv} \subset C$ such that $\langle u, v \mid S'_{uv} \rangle_{\mathcal{G}}^{sep}$,*
3. *Vertices u and v are contained together in some node C and both of them belong to some separator connected to C, but there is a subset S'_{uv} of either $\bigcup_{u \in C'} C'$ or $\bigcup_{v \in C'} C'$ such that $\langle u, v \mid S'_{uv} \rangle_{\mathcal{G}}^{sep}$.*

Skeleton Recovery Algorithm

Algorithm 1 (LCD) Skeleton Recovery

Input: A separation tree $\mathcal{T}(G, \mathcal{C})$; perfect conditional independence knowledge about $\mathbb{P}$.

Output: The skeleton G' of G; a set $\mathcal{S}$ of c-separators.

procedure RECOVERYSKELETON($\mathcal{T}(G, \mathcal{C})$)
  $\mathcal{S} = \emptyset$
  for all node $C_h \in \mathcal{T}(G, \mathcal{C})$ **do**
    Create complete undirected graph $G_h = (C_h, E_h)$;
    for all vertex pair $\{u, v\} \subset C_h$ **do**
      if $\exists S_{uv} \subset C_h \; u \perp\!\!\!\perp v \mid S_{uv}$ **then**
        Delete edge (u, v) from graph G_h;
        $\mathcal{S} := \mathcal{S} \cup S_{uv}$; ▷ Add set S_{uv} to separators
      end if
    end for
  end for
  Combine all the graphs $(G_h)_{i \in \{1,\dots,H\}}$ into undirected graph $G' = (V, \bigcup_{h=1}^{H} E_h)$;

Skeleton Recovery Algorithm

for all $\{u, v\} \in G'$ contained in more then one node of $\mathcal{T}(G, \mathcal{C})$ **do**
    if $\exists C_h\ \{u, v\} \subset C_h$ and $(u, v) \notin E_h$ **then**
        Delete the edge (u, v) from G';
    end if
end for
for all $\{u, v\} \in G'$ contained in more then one node of $\mathcal{T}(G, \mathcal{C})$ **do**
    $N_{uv} := \{S \subset \mathrm{ne}_{G'}(u) \cup \mathrm{ne}_{G'}(v) \mid S \not\subset C_h \text{ and } \{u, v\} \subset C_h\}$
    if $u \perp\!\!\!\perp v \mid S_{uv}$ for some $S_{uv} \subset N_{uv}$ **then**
        Delete edge (u, v) from graph G';
        $\mathcal{S} := \mathcal{S} \cup S_{uv}$; $\triangleright$ Add set S_{uv} to separators
    end if
end for
return: G', $\mathcal{S}$.
end procedure

Complex Recovery

Theorem. *Let G be a chain graph and $\mathcal{T}(G, \mathcal{C})$ be a separation tree of G. For any complex $\mathcal{K}$ in G, there exists some node tree C of $\mathcal{T}(G, \mathcal{C})$ such that $\mathcal{K} \subset C$.*

Complex Recovery Algorithm

Algorithm 2 (LCD) Complex Recovery

Input: Perfect conditional independence knowledge about $\mathbb{P}$; the skeleton G' and the set $\mathcal{S}$ of c-separators obtained in algorithm 1.

Output: The pattern G^* of graph G.

procedure $\textsc{ComplexRecovery}(\mathcal{T}(G, \mathcal{C}))$
    Initialize $G^* = G'$
    for all ordered pair $[u, v]\ : S_{uv} \in \mathcal{S}$ **do**
        for all $u - w$ in G^* **do**
            if $u \not\perp\!\!\!\perp v \mid S_{uv} \cup \{w\}$ **then**
                Orient $u - w$ as $u \to w$ in G^*;
            end if
        end for
    end for
    return: Pattern of G^*.
end procedure

Testing conditional independence

Under assumption of conditional independence $B \perp\!\!\!\perp C \mid A$ we have

$$
\begin{aligned}
&\mathbb{P}(A = i, B = j, C = k) = \\
&\quad \mathbb{P}(A = i)\mathbb{P}(B = j, C = k \mid A = i) = \\
&\quad \mathbb{P}(A = i)\mathbb{P}(B = j \mid A = i)\mathbb{P}(C = k \mid A = i) = \\
&\quad p_{i++}p_{j|i}p_{k|i}
\end{aligned}
$$

$$
\hat{p_{i++}} = \frac{N_{i++}}{n}
$$

$$
\hat{p_{j|i}} = \frac{N_{ij+}}{N_{i++}}
$$

$$
\hat{p_{k|i}} = \frac{N_{i+k}}{N_{i++}}
$$

Testing conditional independence

Using G-statistic $G^2 = 2\sum_{i=1}^{n} O_i \ln\left(\frac{O_i}{E_i}\right) \sim \chi^2(n)$ we have test statistic

$$G^2 = 2\sum_{i,j,k} N_{ijk} \ln\left(\frac{N_{ij+}N_{i+k}}{N_{ijk}N_{i++}}\right)$$

In this case $G^2 \sim \chi^2(\#(B-1)\#(C-1)\#A)$