Conditional Independencies

+ Factor Graphs

Matt Gormley
Lecture 7
Sep. 23, 2022
Reminders

• Homework 2: Learning to Search for RNNs
  – Out: Sun, Sep 18
  – Due: Thu, Sep 29 at 11:59pm
Q: What is the difference between a stochastic policy and a deterministic policy?

A: Definition: a **stochastic policy** is a probability distribution over actions given a state

\[ a_t \sim \pi(\cdot \mid s_t) \]

Definition: a **deterministic policy** is a function that maps from a state to an action

\[ a_t = \pi(s_t) \]
CONDITIONAL INDEPENDENCIES OF DGMS
What Independencies does a Bayes Net Model?

Three cases of interest...

**Cascade**

- Z
- Y
- X

**Common Parent**

- Y
- X → Z

**V-Structure**

- X → Y
- Y → Z
What Independencies does a Bayes Net Model?

Three cases of interest...

**Cascade**

![Diagram](cascade.png)

Knowing Y **decouples** X and Z

**Common Parent**

![Diagram](common_parent.png)

Knowing Y **couples** X and Z

**V-Structure**

![Diagram](v_structure.png)
D-Separation

**If** variables X and Z are **d-separated** given a **set** of variables E

**Then** X and Z are **conditionally independent** given the **set** E

**Definition #1:**
Variables X and Z are **d-separated** given a **set** of evidence variables E iff every path from X to Z is “blocked”.

A path is “blocked” whenever:

1. \( \exists Y \) on path s.t. \( Y \in E \) and \( Y \) is a “common parent”

\[
\begin{array}{c}
  X \rightarrow \ldots \rightarrow \bigcirc \leftrightarrow Y \rightarrow \bigcirc \rightarrow \ldots \rightarrow Z
\end{array}
\]

2. \( \exists Y \) on path s.t. \( Y \in E \) and \( Y \) is in a “cascade”

\[
\begin{array}{c}
  X \rightarrow \ldots \rightarrow \bigcirc \leftrightarrow Y \rightarrow \bigcirc \rightarrow \ldots \rightarrow Z
\end{array}
\]

3. \( \exists Y \) on path s.t. \{Y, descendants(Y)\} \( \notin E \) and \( Y \) is in a “v-structure”

\[
\begin{array}{c}
  X \rightarrow \ldots \rightarrow \bigcirc \leftrightarrow Y \rightarrow \bigcirc \rightarrow \ldots \rightarrow Z
\end{array}
\]
D-Separation

If variables X and Z are **d-separated** given a **set** of variables E
Then X and Z are **conditionally independent** given the set E

**Definition #2:**
Variables X and Z are **d-separated** given a **set** of evidence variables E iff there does not exist a path in the **undirected ancestral moral** graph with E removed.

1. **Ancestral graph:** keep only X, Z, E and their ancestors
2. **Moral graph:** add undirected edge between all pairs of each node’s parents
3. **Undirected graph:** convert all directed edges to undirected
4. **Givens Removed:** delete any nodes in E

**Example Query:**  $A \perp B \mid \{D, E\}$

Original:  
Ancestral:  
Moral:  
Undirected:  
Givens Removed:  

$\Rightarrow A$ and $B$ connected  
$\Rightarrow$ not d-separated
Markov Boundary (Directed)

**Def:** the co-parents of a node are the parents of its children.

**Def:** the Markov boundary of a node is the set containing the node’s parents, children, and co-parents.
Def: the co-parents of a node are the parents of its children

Def: the Markov boundary of a node is the set containing the node’s parents, children, and co-parents.

Example: The Markov boundary of $X_6$ is \{X_3, X_4, X_5, X_8, X_9, X_{10}\}
Markov Boundary (Directed)

**Def:** the co-parents of a node are the parents of its children

**Def:** the Markov boundary of a node is the set containing the node’s parents, children, and co-parents.

**Theorem:** a node is conditionally independent of every other node in the graph given its Markov boundary

**Example:** The Markov boundary of $X_6$ is $\{X_3, X_4, X_5, X_8, X_9, X_{10}\}$
CONDITIONAL INDEPENDENCIES
OF UGMS
Markov Boundary (Directed)

**Def:** the **co-parents** of a node are the parents of its children

**Def:** the **Markov boundary** of a node is the set containing the node’s parents, children, and co-parents.

**Theorem:** a node is conditionally independent of every other node in the graph given its **Markov boundary**

**Example:** The Markov boundary of $X_6$ is \{$X_3, X_4, X_5, X_8, X_9, X_{10}$\}
**Markov Boundary (Undirected)**

**Def:** the **Markov boundary** of a node in an **undirected** graphical model is the set containing the node’s neighbors.

**Theorem:** a node is conditionally independent of every other node in the graph given its **Markov boundary**

**Example:** The Markov boundary of $X_6$ is \{X_3, X_4, X_9, X_{10}\}
Q: What is the difference between a Markov boundary and Markov blanket?

A: In a graphical model, the **Markov blanket** for X is *any* set S s.t. X is conditionally independent of all other variables when conditioned on S.

The **Markov boundary** for X is the smallest possible Markov blanket.

Every Markov boundary is a Markov blanket but not vice versa.
Undirected Graphical Models

**Conditional Independence Semantics**

Consider a distribution over r.v.s $X_1, \ldots, X_T$

For a UGM and any disjoint index sets $A$, $B$, $C$, (i.e., $A \subseteq \{1, \ldots, T\}$, $B \subseteq \{1, \ldots, T\}$, $C \subseteq \{1, \ldots, T\}$)

$X_A$ is **conditionally independent** of $X_B$ given $X_C$ iff $X_C$ separates sets $X_A$ and $X_B$
Undirected Graphical Models

Whiteboard

- Proof of independence by separation (simple case)
Non-equivalence of Directed / Undirected Graphical Models

There does not exist an **undirected** graphical model that can capture the conditional independence assumptions of this **directed** graphical model:

There does not exist a **directed** graphical model that can capture the conditional independence assumptions of this **undirected** graphical model:

**Exercise:** Can you prove these claims?
Representation of both directed and undirected graphical models

FACTOR GRAPHS
Three Types of Graphical Models

Directed Graphical Model

Undirected Graphical Model

Factor Graph
How General Are Factor Graphs?

• Factor graphs can be used to describe
  – Markov Random Fields (undirected graphical models)
  – Conditional Random Fields
  – Bayesian Networks (directed graphical models)
Factor Graph Notation

• Variables:
\[ \mathcal{X} = \{X_1, \ldots, X_i, \ldots, X_n\} \]

• Factors:
\[ \psi_\alpha, \psi_\beta, \psi_\gamma, \ldots \]
where \( \alpha, \beta, \gamma, \ldots \subseteq \{1, \ldots n\} \)

Joint Distribution

\[ p(x) = \frac{1}{Z} \prod_\alpha \psi_\alpha(x_\alpha) \]
Factors are Tensors

- **Def**: the *arity* of a factor is the number of neighbors (variables) it has.

- **Factors:**
  \[ \psi_\alpha, \psi_\beta, \psi_\gamma, \ldots \]
  where \( \alpha, \beta, \gamma, \ldots \subseteq \{1, \ldots n\} \)

- **Def**: a *unary factor* touches one variable.
- **Def**: a *binary factor* touches two variables.
- **Def**: a *ternary factor* touches three variables.
Factors are Tensors

- Factors must contain non-negative values -- this ensures we have a valid probability distribution.
- We also sometimes refer to factors as potential functions or potentials (like UGMs).

Joint Distribution

\[ p(x) = \frac{1}{Z} \prod_{\alpha} \psi_{\alpha}(x_{\alpha}) \]
Ex: Factor Graph over Binary Variables

\[
P(A=a, B=b, C=c) = p(a, b, c) = \frac{1}{Z} \psi_A(a) \psi_{AB}(a, b) \psi_{ABC}(a, b, c)
\Rightarrow Z = \sum_{a} \sum_{b} \sum_{c} s(a, b, c)
\]

<table>
<thead>
<tr>
<th>a \ b \ c</th>
<th>\psi_A</th>
<th>\psi_{AB}</th>
<th>\psi_{ABC}</th>
<th>s(\cdot)</th>
<th>p(\cdot)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0</td>
<td>2 4 6</td>
<td></td>
<td></td>
<td>48</td>
<td>48/Z</td>
</tr>
<tr>
<td>0 0 1</td>
<td>2 4 2</td>
<td></td>
<td></td>
<td>16</td>
<td>16/Z</td>
</tr>
<tr>
<td>0 1 0</td>
<td>2 3 1</td>
<td></td>
<td></td>
<td>6</td>
<td>6/Z</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td></td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>1 1 1</td>
<td>7 2 5</td>
<td></td>
<td>+ 70</td>
<td>70</td>
<td>70/Z</td>
</tr>
</tbody>
</table>

\[
Z
\]
CONVERTING UGMS AND DGMS TO FACTOR GRAPHS
Factors have local opinions ($\geq 0$)

Each black box looks at some of the tags $X_i$ and words $W_i$

Note: We chose to reuse the same factors at different positions in the sentence.
Factors have local opinions (≥ 0)

Each black box looks at some of the tags $X_i$ and words $W_i$

$$p(n, v, p, d, n, \text{time, flies, like, an, arrow}) = ?$$
Global probability = product of local opinions

Each black box looks at some of the tags $X_i$ and words $W_i$

$$p(n, v, p, d, n, \text{time}, \text{flies}, \text{like}, \text{an}, \text{arrow}) = \frac{1}{Z} (4 * 8 * 5 * 3 * \ldots)$$

Uh-oh! The probabilities of the various assignments sum up to $Z > 1$. So divide them all by $Z$. 
Markov Random Field (MRF)

Joint distribution over tags $X_i$ and words $W_i$
The individual factors aren’t necessarily probabilities.

$$p(n, v, p, d, n, \text{time, flies, like, an, arrow}) = \frac{1}{\mathcal{Z}}(4 \times 8 \times 5 \times 3 \times \ldots)$$
Bayesian Networks

But sometimes we choose to make them probabilities. Constrain each row of a factor to sum to one. Now $Z = 1$.

$$p(n, v, p, d, n, \text{time, flies, like, an, arrow}) = \frac{1}{Z} (.3 \times .8 \times .2 \times .5 \times \ldots)$$
Converting to Factor Graphs

Each conditional and marginal distribution in a **directed GM** becomes a factor

Each maximal clique (or each clique) in an **undirected GM** becomes a factor
Converting to Factor Graphs

• Example 1
Converting to Factor Graphs

• Example 2

\[ \psi(x_1, x_2, x_3) = f_a(x_1, x_2) f_b(x_2, x_3) f_c(x_3, x_1) \]

• Example 3

\[ \psi(x_1, x_2, x_3) = f_a(x_1, x_2, x_3) \]
Converting to Factor Graphs

• Example 4
Converting to Factor Graphs

• Example 5
Converting to Factor Graphs

A neat property of factor graph conversion:

- A factor graph sometimes turns tree-like undirected / directed graphical models to factor trees,
- Trees are a data-structure that guarantees exactness of belief propagation!
Converting to Factor Graphs

Equivalence of directed and undirected trees

• Any undirected tree can be converted to a directed tree by choosing a root node and directing all edges away from it

• A directed tree and the corresponding undirected tree make the same conditional independence assertions

• Parameterizations are essentially the same.

  – Undirected tree:

\[
p(x) = \frac{1}{Z} \left( \prod_{i \in V} \psi(x_i) \prod_{(i,j) \in E} \psi(x_i, x_j) \right)
\]

  – Directed tree:

\[
p(x) = p(x_r) \prod_{(i,j) \in E} p(x_j | x_i)
\]

  – Equivalence:

\[
\psi(x_r) = p(x_r); \quad \psi(x_i, x_j) = p(x_j | x_i);
\]

\[
Z = 1, \quad \psi(x_i) = 1
\]
MRF VS. CRF
MRF vs. CRF

Markov Random Field (MRF):
- just a distribution over variables $y$
- partition function $Z$ is just a function of the parameters

\[
p_{\theta}(y) = \frac{1}{Z(\theta)} \prod_{\alpha} \psi_{\alpha}(y_\alpha; \theta)
\]

Conditional Random Field (CRF):
- conditions on some additional observed variables $x$
- partition function $Z$ is a function of $x$ as well

\[
p_{\theta}(y \mid x) = \frac{1}{Z(x; \theta)} \prod_{\alpha} \psi_{\alpha}(y_\alpha, x; \theta)
\]
Markov Random Field (MRF)

Joint distribution over tags $X_i$ and words $W_i$

$$p(n, v, p, d, n, \text{time, flies, like, an, arrow}) = \frac{1}{Z} (4 \times 8 \times 5 \times 3 \times \ldots)$$
Conditional Random Field (CRF)

Conditional distribution over tags $X_i$ given words $w_i$. The factors and Z are now specific to the sentence $w$.

$$p(n, v, p, d, n \mid \text{time, flies, like, an, arrow}) = \frac{1}{Z} (4 \times 8 \times 5 \times 3 \times \ldots)$$
TYPES OF GRAPHICAL MODELS
Three Types of Graphical Models

Directed Graphical Model

Undirected Graphical Model

Factor Graph
Key Concepts for Graphical Models

**Graphical Models in General**

1. A graphical model defines a family of probability distributions
2. That family shares in common a set of conditional independence assumptions
3. By choosing a parameterization of the graphical model, we obtain a single model from the family
4. The model may be either locally or globally normalized

**Ex: Directed G.M.**

1. **Family**: directed graphs with locally normalized conditional probabilities
2. **Conditional Dependencies**: d-separation, Markov blanket
3. **Example parameterization**: conditional probability tables (CPTs) for discrete var.s, conditional probability densities for continuous var.s
4. **Normalization**: locally normalized, partition function is always 1.0
Key Concepts for Graphical Models

Graphical Models in General
1. A graphical model defines a family of probability distributions
2. That family shares in common a set of conditional independence assumptions
3. By choosing a parameterization of the graphical model, we obtain a single model from the family
4. The model may be either locally or globally normalized

Ex: Undirected G.M.
1. Family: undirected graphs with unnormalized potentials
2. Conditional Independencies: independence by separation, Markov blanket
3. Example parameterization: Markov random field (MRF), conditional random field (CRF), neural potentials
4. Normalization: globally normalized
Key Concepts for Graphical Models

Graphical Models in General
1. A graphical model defines a family of probability distributions
2. That family shares in common a set of conditional independence assumptions
3. By choosing a parameterization of the graphical model, we obtain a single model from the family
4. The model may be either locally or globally normalized

Ex: Factor Graph
1. Family: bipartite graph over variables and factors
2. Conditional Independencies: independence by separation, inferable from underlying DGM or UGM
3. Example parameterization: any DGM parameterization, any UGM parameterization
4. Normalization: locally normalized if based on DGM, globally normalized if based on UGM
Q&A
COMPUTATIONAL COMPLEXITY
Analysis of Algorithms

Key Questions:
1. Given a single algorithm, will it complete on a given input in a reasonable amount of time/space?
2. Given two algorithms, which one is better?
Comparing Algorithm Runtimes

- $O(1)$
- $O(\log(n))$
Comparing Algorithm Runtimes

\[ O(1) \]
\[ O(\log(n)) \]
\[ O(n) \]
Comparing Algorithm Runtimes

- $O(1)$
- $O(\log(n))$
- $O(n)$
- $O(n\log(n))$
Comparing Algorithm Runtimes

Graph showing the runtimes of different algorithms:
- $O(1)$
- $O(\log(n))$
- $O(n)$
- $O(n \log(n))$
- $O(n^2)$

The graph plots runtime on the y-axis against input size on the x-axis.
Comparing Algorithm Runtimes

- $O(1)$
- $O(\log(n))$
- $O(n)$
- $O(n \log(n))$
- $O(n^2)$
- $O(n^3)$
Comparing Algorithm Runtimes

- $O(1)$
- $O(\log(n))$
- $O(n)$
- $O(n\log(n))$
- $O(n^2)$
- $O(n^3)$
- $O(2^n)$
Comparing Algorithm Runtimes

$O(1)$

$O(\log(n))$

$O(n)$

$O(n \log(n))$

$O(n^2)$

$O(n^3)$

$O(2^n)$
Comparing Algorithm Runtimes

The graph compares the runtimes of various algorithm complexities. The x-axis represents the input size, while the y-axis shows the runtime in arbitrary units. The complexities shown include:

- \( O(1) \) (constant time)
- \( O(\log(n)) \) (logarithmic time)
- \( O(n) \) (linear time)
- \( O(n\log(n)) \) (quasilinear time)
- \( O(n^2) \) (quadratic time)
- \( O(n^3) \) (cubic time)
- \( O(2^n) \) (exponential time)

As the input size increases, the exponential runtime becomes significantly higher than the linear or even quadratic runtimes.
# Comparing Algorithm Runtimes

<table>
<thead>
<tr>
<th>Computational Complexity</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>( O(1) )</td>
<td>constant</td>
</tr>
<tr>
<td>( O(\log(n)) )</td>
<td>logarithmic</td>
</tr>
<tr>
<td>( O(n) )</td>
<td>linear</td>
</tr>
<tr>
<td>( O(n \log(n)) )</td>
<td>“n log n”</td>
</tr>
<tr>
<td>( O(n^2) )</td>
<td>quadratic</td>
</tr>
<tr>
<td>( O(n^3) )</td>
<td>cubic</td>
</tr>
<tr>
<td>( O(2^n) )</td>
<td>exponential</td>
</tr>
<tr>
<td>( O(n!) )</td>
<td>factorial</td>
</tr>
<tr>
<td>( O(n^n) )</td>
<td>superexponential</td>
</tr>
</tbody>
</table>
Complexity Classes

- An algorithm runs in **polynomial time** if its runtime is a polynomial function of the input size (e.g. $O(n^k)$ for some fixed constant $k$)
- The **class P** consists of all problems that can be solved in polynomial time

- A problem for which the answer is binary (e.g. yes/no) is called a **decision problem**
- The **class NP** contains all decision problems where ‘yes’ answers can be verified (proved) in polynomial time
- A problem is **NP-Hard** if given an $O(1)$ oracle to solve it, every problem in NP can be solved in polynomial time (e.g. by reduction)
- A problem is **NP-Complete** if it belongs to both the classes NP and NP-Hard

Figure from https://en.wikipedia.org/wiki/NP-completeness
Complexity Classes

• A problem for which the answer is a nonnegative integer is called a **counting problem**

• The **class #P** contains the counting problems that align to decision problems in NP
  – really this is the class of problems that count the number of accepting paths in a Turing machine that is nondeterministic and runs in polynomial time

• A problem is **#P-Hard** if given an O(1) oracle to solve it, every problem in #P can be solved in polynomial time (e.g. by reduction)

• A problem is **#P-Complete** if it belongs to both the classes #P and #P-Hard

• There are no known polytime algorithms for solving #P-Complete problems. If we found one it would imply that P = NP.

**Examples of #P-Hard problems**

• #SAT, i.e. how many satisfying solutions for a given SAT problem?
• How many solutions for a given DNF formula?
• How many solutions for a 2-SAT problem?
• How many perfect matchings for a bipartite graph?
• How many graph colorings (with k colors) for a given graph G?

Examples from https://en.wikipedia.org/wiki/%E2%99%AFP-complete
EXACT INFERENCE
Exact Inference

1. Data

\[ D = \{ x^{(n)} \}_{n=1}^{N} \]

Sample 1:
- time
- dies
- an
- arrow

Sample 2:
- time
- dies
- an
- arrow

Sample 3:
- flies
- with
- their
- wings

Sample 4:
- time
- like
- flies
- an
- arrow

2. Model

\[ p(x \mid \theta) = \frac{1}{Z(\theta)} \prod_{C \in \mathcal{C}} \psi_C(x_C) \]

3. Objective

\[ \ell(\theta; D) = \sum_{n=1}^{N} \log p(x^{(n)} \mid \theta) \]

4. Learning

\[ \theta^* = \arg\max_{\theta} \ell(\theta; D) \]

5. Inference

1. Marginal Inference

\[ p(x_C) = \sum_{x' : x'_C = x_C} p(x' \mid \theta) \]

2. Partition Function

\[ Z(\theta) = \sum_{x} \prod_{C \in \mathcal{C}} \psi_C(x_C) \]

3. MAP Inference

\[ \hat{x} = \arg\max_x p(x \mid \theta) \]
5. Inference

Three Tasks:

1. **Marginal Inference (#P-Hard)**
   Compute marginals of variables and cliques
   \[
   p(x_i) = \sum_{x': x'_i = x_i} p(x' | \theta)
   \]
   \[
   p(x_C) = \sum_{x': x'_C = x_C} p(x' | \theta)
   \]

2. **Partition Function (#P-Hard)**
   Compute the normalization constant
   \[
   Z(\theta) = \sum_x \prod_{C \in C} \psi_C(x_C)
   \]

3. **MAP Inference (NP-Hard)**
   Compute variable assignment with highest probability
   \[
   \hat{x} = \arg\max_x p(x | \theta)
   \]