15-780 – Probabilistic Inference

J. Zico Kolter

March 30, 2014
Outline

Probabilistic graphical models

Exact inference

Approximate inference
Outline

Probabilistic graphical models

Exact inference

Approximate inference
Probabilistic graphical models (PGMs) are about representing probability distributions over random variables

\[ p(x) \]

where for this lecture, \( x \in \{0, 1\}^n, p : \{0, 1\}^n \rightarrow [0, 1] \)

- Naively, since there are \( 2^n \) possible assignments to \( x \), can represent this distribution completely using \( 2^n - 1 \) numbers, but quickly becomes intractable for large \( n \)

- PGMs are methods to represent these distributions more compactly, by exploiting *conditional independence*
Bayesian networks

- A Bayesian network is defined by:

  1. A directed acyclic graph (DAG) $G = (V = \{x_1, \ldots, x_n\}, E)$

  2. A set of conditional probability tables $p(x_i|\text{Parents}(x_i))$

- Defines the joint probability distribution

  $$p(x) = \prod_{i=1}^{n} p(x_i|\text{Parents}(x_i))$$

- Equivalently, each node is conditionally independent of all non-descendants given its parents
Bayes net example

Burglary? Earthquake?

JohnCalls? MaryCalls?

Can write distribution as

\[ p(x) = p(x_1) p(x_2 | x_1) p(x_3 | x_1, x_2) p(x_4 | x_3) p(x_5 | x_3) \]
Bayes net example

- $p(x_1 = 1) = 0.001$
- $p(x_2 = 1) = 0.002$
- $p(x_3 = 1)$

<table>
<thead>
<tr>
<th>$x_3$</th>
<th>$p(x_4 = 1)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.05</td>
</tr>
<tr>
<td>1</td>
<td>0.9</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$x_3$</th>
<th>$p(x_5 = 1)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.01</td>
</tr>
<tr>
<td>1</td>
<td>0.7</td>
</tr>
</tbody>
</table>

- Burglary? $x_1$
- Earthquake? $x_2$
- Alarm? $x_3$
- JohnCalls? $x_4$
- MaryCalls? $x_5$

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$p(x_3 = 1)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0.001</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0.29</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0.94</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0.95</td>
</tr>
</tbody>
</table>
Bayes net example

- Can write distribution as

\[ p(x) = p(x_1)p(x_2|x_1)p(x_3|x_1, x_2)p(x_4|x_1, x_2, x_3)p(x_5|x_1, x_2, x_3, x_4) \]

\[ = p(x_1)p(x_2)p(x_3|x_1, x_2)p(x_4|x_3)p(x_5|x_3) \]
Markov random fields

- A (pairwise) Markov random field (MRF) is defined by:
  1. An undirected graph $G = (V = \{x_1, \ldots, x_n\}, E)$
  2. A set of unary potential $f(x_i)$ for each $i = 1, \ldots, n$
  3. A set of binary potentials $f(x_i, x_j)$ for all $i, j \in E$

- Defines the joint probability distribution

$$p(x) = \frac{1}{Z} \prod_{i=1}^{n} f(x_i) \prod_{i,j \in E} f(x_i, x_j)$$

where $Z$ is a normalization constant (also called partition function)

$$Z = \sum_{x} \prod_{i=1}^{n} f(x_i) \prod_{i,j \in E} f(x_i, x_j)$$
• Equivalently, each node is in MRF is conditionally independent of all other nodes given it’s neighbors

\[ p(x_i|x_{-i}) = p(x_i|\text{Neighbors}(x_i)) \]

not trivial to show, known as Hammersley-Clifford theorem
MRF example

\[ p(x_1 = 1, x_2 = 1) = \frac{1}{3} \]
MRF example

\[
\begin{array}{|c|c|c|}
\hline
x_1 & x_2 & f(x_1, x_2) \\
\hline
0 & 0 & 10 \\
0 & 1 & 1 \\
1 & 0 & 1 \\
1 & 1 & 10 \\
\hline
\end{array}
\]
E.g. $p(x_1 = 1, x_2 = 1) = \frac{1}{150} \cdot 5 \cdot 10 \cdot 1 = 1/3$
Factor graphs

• A generalization that captures both Bayesian networks and Markov random fields

• An undirected graph, $G = \{V = \{x_1, \ldots, x_n, f_1, \ldots, f_m\}, E\}$ over variables and factors

• There exists an edge $f_i \rightarrow x_j$ if and only if factor $f_i$ includes variable $x_j$

• Defines the joint probability distribution

$$p(x) = \frac{1}{Z} \prod_{i=1}^{m} f_i(x_i)$$

where $x_i = \{x_j : (f_i, x_j) \in E\}$ are all variables in factor $f_i$
MRF to factor graph

\[ x_1 \quad \rightarrow \quad x_2 \]
MRF to factor graph

\[ x_1 \quad f_3 \quad x_2 \]

\[ f_1 \quad f_2 \]
MRF to factor graph

\[ f_3(x_1, x_2) \]

<table>
<thead>
<tr>
<th>(x_1)</th>
<th>(x_2)</th>
<th>(f_3(x_1, x_2))</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>10</td>
</tr>
</tbody>
</table>

![Graph representation of MRF to factor graph]
Bayes net to factor graph
Bayes net to factor graph

\[ x_1 \quad f_1 \quad x_3 \quad f_3 \quad x_2 \quad f_2 \]

\[ x_3 \quad f_4 \quad f_5 \quad x_4 \quad x_5 \]
Bayes net to factor graph

\[
x_1 \quad f_1 \\
f_3 \\
x_2 \quad f_2 \\
f_4 \quad f_5 \\
x_3 \\
x_4 \\
x_5
\]

\[
\begin{array}{c|c|c}
  x_3 & p(x_5 = 1) \\
  \hline
  0 & 0.01 \\
  1 & 0.7 \\
\end{array}
\]
Bayes net to factor graph

\[ p(x_5 = 1) \]

\[
\begin{array}{c|c|c}
 x_3 & x_5 & f_5(x_3, x_5) \\
 \\
 0 & 0 & 0.99 \\
 0 & 1 & 0.01 \\
 1 & 0 & 0.3 \\
 1 & 1 & 0.7 \\
\end{array}
\]
Outline

Probabilistic graphical models

Exact inference

Approximate inference
Inference in probabilistic graphical models

- **Inference** generally refers to methods that query probabilities given a graphical model.

- Several types that come up frequently
  - *Marginal inference*: compute \( p(x_I) \) for some \( x_I \subseteq \{x_1, \ldots, x_n\} \) (non-trivial even for \( x_I = \{x_1, \ldots, x_n\} \) in factor graph).
  
  - *Conditional inference*: compute \( p(x_I | x_E = x_E^0) \) for some \( x_I, x_E \subseteq \{x_1, \ldots, x_n\}, x_I \cap x_E = \emptyset \).
  
  - *Maximum a posteriori (MAP) inference*: compute \( \max_{x_I} p(x_I) \), and possibly the maximizing assignment \( x_I^* \) (also, conditional analogue); also called most probable explanation (MPE).
Inference via enumeration

• If we’re willing to enumerate all $2^n$ possible values, inference queries can be answered easily
  
  – Marginal inference:

  \[
  p(x_I) = \sum_{\bar{x}_I} p(x_I, \bar{x}_I) = \sum \prod_{i=1}^{m} f_i(\mathcal{X}_i)
  \]

  – Conditional inference

  \[
  p(x_I | x_E = x_0^E) = \frac{p(x_I, x_E = x_0^E)}{p(x_E = x_0^E)}
  \]

  – MAP inference: compute $p(x_I = x_0^I)$ for all possible assignments $x_0^I$, choose largest
Exploiting graph structure in inference

- When $n$ gets large, inference by exact enumeration is intractable
- Can (sometimes) use compact graph representation of the distribution to derive compact forms of inference
Example: chain Bayesian network

\[ p(x_4) = \sum_{x_1, x_2, x_3} p(x_1, x_2, x_3, x_4) \]
Example: chain Bayesian network

\[ p(x_4) = \sum_{x_1,x_2,x_3} p(x_1)p(x_2|x_1)p(x_3|x_2)p(x_4|x_3) \]
Example: chain Bayesian network

$$p(x_4) = \sum_{x_2, x_3} p(x_3 | x_2)p(x_4 | x_3) \sum_{x_1} p(x_1)p(x_2 | x_1)$$
Example: chain Bayesian network

\[ p(x_4) = \sum_{x_2, x_3} p(x_3 | x_2) p(x_4 | x_3) p(x_2) \]
Example: chain Bayesian network

\[ p(x_4) = \sum_{x_3} p(x_4|x_3) \sum_{x_2} p(x_3|x_2)p(x_2) \]
Example: chain Bayesian network

\[ p(x_4) = \sum_{x_3} p(x_4 | x_3) p(x_3) = p(x_4) \]
General algorithm: variable elimination

function $G' = \text{Sum-Product-Eliminate}(G, x_i)$

// eliminate variable $x_i$ from the factor graph $G$

$F \leftarrow \{ f_j \in V : (f_j, x_i) \in E \}$

$\tilde{X} \leftarrow \{ x_k : (f_j, x_k) \in E, f_j \in F \} - \{ x_i \}$

$\tilde{f}(\tilde{X}) \leftarrow \sum x_i \prod_{f_j \in F} f_j(x_j)$

$V' \leftarrow V - \{ x_i, f_j \in F \} + \{ \tilde{f} \}$

$E' \leftarrow E - \{ (f_j, x_k) \in E : f_j \in F \} + \{ (\tilde{f}, x_k) : x_k \in \tilde{X} \}$

return $G' = (V', E')$
Variable elimination example
Variable elimination example

\[ F = \{ f_3, f_4, f_5 \} \]
Variable elimination example

\[ F = \{ f_3, f_4, f_5 \} \]
\[ \tilde{X} = \{ x_1, x_2, x_4, x_4 \} \]
Variable elimination example

\[ \tilde{f}(x_1, x_2, x_3, x_5) = \sum_{x_3} f_3(x_1, x_2, x_3)f_4(x_3, x_4)f_5(x_3, x_5) \]

\[ F = \{ f_3, f_4, f_5 \} \]
\[ \tilde{X} = \{ x_1, x_2, x_4, x_5 \} \]
\[ \tilde{f} = \{ (f_1, x_1), (f_2, x_2), (\tilde{f}, x_1), (\tilde{f}, x_2), (\tilde{f}, x_4), (\tilde{f}, x_5) \} \]
• Full variable elimination algorithm just repeatedly eliminates variables

```plaintext
function \( G' = \text{Sum-Product-Variable-Elimination}(G, X) \)
// eliminate an ordered list of variables \( X \)
for \( x_i \in X \):
    \( G \leftarrow \text{Sum-Product-Eliminate}(G, x_i) \)
return \( G \)
```

• Graph returned at the end is a marginalized factor graph over non-eliminated variables (eliminating all variables returns constant equal to partition function \( Z \))

• The ordering matters a lot, eliminating variables in the wrong order can make algorithm no better than enumeration
Variable elimination example

Goal: compute \( p(x_4) \)
Variable elimination example

Goal: compute $p(x_4)$
Variable elimination example

Goal: compute $p(x_4)$
Variable elimination example

Goal: compute $p(x_4)$
Variable elimination example

Goal: compute $p(x_4)$
Variable elimination example

Goal: compute $p(x_4)$
Variable elimination example

Goal: compute $p(x_4)$
Variable elimination example

Goal: compute $p(x_4)$
Variable elimination example

Goal: compute $p(x_4)$
Pitfalls

- **Tree-width** of graphical model is size of the maximum factor formed during variable elimination (assuming best ordering); inference is exponential in tree width

- But...
  - Finding best variable elimination ordering is NP-hard
  - Some “simple” graphs have high tree-width (e.g., $M \times N$ “grid” MRF has tree-width $\min(M, N)$)
Extensions

• Difficulty with variable elimination as stated is that we need to “rerun” algorithm each time we want to make an inference query.

• Solution: slight extension of variable elimination that caches intermediate factors, making a forward and backward pass over all variables (Junction Tree or Clique Tree algorithm).

• You’ll probably see these algorithms written in terms of message passing, but these “messages” are just intermediate factors \( \hat{f} \)’s.
MAP Inference

- Virtually identical approach can be applied to MAP inference

- Only change is replacing sum-product operation

\[ \tilde{f}(\tilde{X}) \leftarrow \sum_{x_i} \prod_{f_j \in F} f_j(X_j) \]

with max-product operation

\[ \tilde{f}(\tilde{X}) \leftarrow \max_{x_i} \prod_{f_j \in F} f_j(X_j) \]

- If we want to find actual maximizing assignment, also need to keep a separate function of which \( x_i \) value is maximal for each \( \tilde{f}(X) \)
Outline

Probabilistic graphical models

Exact inference

Approximate inference
Sampling methods

• Instead of exactly computing probabilities $p(x)$, we may want to draw random samples from this distribution $x \sim p(x)$

• For example, in Bayesian networks this is straightforward, just sample individual variables sequentially

\[ x_i \sim p(x_i | \text{Parents}(x_i)) \quad i = 1, \ldots, n \]

• For cases where we can efficiently perform variable elimination, a slightly modified procedure lets us draw random samples (perhaps conditioned on evidence)
Gibbs sampling

• But what about cases too big for variable elimination?

• A common solution: Gibbs sampling

\[
\text{function } x = \text{Gibbs-Sampling}(G, x, K) \\
\text{for } i = 1, \ldots, K: \\
\text{Choose a random } x_i \\
\text{Sample } x_i \sim p(x_i | x_{-i}) \propto \prod_{f_j : (f_j, x_i) \in G} f_j(X_j)
\]

• In the limit, \( x \) will be drawn exactly according to the desired distribution (but may take exponentially long to converge)

• One of a broad class of methods called Markov Chain Monte Carlo (MCMC)
Inference as optimization

• Inference in graphical models can be cast as an optimization problem, has been a huge source of ideas for improving exact and approximate inference methods.

• We’re going to consider the simpler case of MAP inference, which already looks like an optimization problem:

$$\max_{x} p(x)$$

• To put this in a form that we’re more familiar with, for each factor $f_i$ define the optimization variable $\mu_i \in \mathbb{R}^{2^{|X_i|}}$; $\mu_i$ should be thought of as an indicator for the assignment to $X_i$. 

• Abusing notation a bit, we can write optimization as a binary integer program

\[
\max_{\mu_1, \ldots, \mu_m} \log p(\mu) = \sum_{i=1}^{m} \mu_i^T (\log f_i)
\]

subject to \( \mu_1, \ldots, \mu_n \) is valid distribution

\((\mu_i)_j \in \{0, 1\}, \ \forall i, j\)

• “Valid distribution” here means assignments have to be consistent, i.e., if \( x_k \in X_i \) and \( x_k \in X_j \), then

\[
\sum_{X_i \setminus \{x_k\}} \mu_i(X_i) = \sum_{X_i \setminus \{x_k\}} \mu_j(X_j)
\]

and they have to have only one non-zero entry \( \sum_j (\mu_i)_j = 1 \)
• This is still a hard, binary integer programming task, but it turns out that the LP relaxation is sometimes tight (i.e., just removing the integer constraints still gives the optimal solution)

• One case where relaxation is tight: tree factor graphs (these are ones we could already solve with max-product)

• Extremely cool: there are other cases where relaxation is still tight even though naive max-product doesn’t apply, like certain grid MRFs

• Can also apply to the case of marginal inference (let $\mu$ terms have non-integer values, but also include terms due to partition function, other constraints)

• A big area of open research
Take home points

• Probabilistic models can compactly represent high dimensional probability distribution

• Inference algorithms provide a method for making probabilistic queries that also (try to) exploit the structure of the distribution

• Wide range of inference methods, ranging from variable elimination for exact inference, sampling and optimization approaches