10-708: Probabilistic Graphical Models 10-708, Spring 2017

Lecture 4: Exact Inference

Lecturer: Eric P. Xing *Scribes: Yohan Jo, Baoyu Jing*

1 Probabilistic Inference and Learning

In practice, exact inference is not used widely, and most probabilistic inference algorithms are approximate. Nevertheless, it is important to understand exact inference and its limitations.

There are two typical tasks with graphical models: inference and learning. Given a graphical model M that describes a unique probability distribution P, inference means answering queries about P_M, e.g., $P_M(X|Y)$. Learning means obtaining a point estimate of model M from data D. However, in statistics, both inference and learning are commonly referred to as either inference or estimation. From the Bayesian perspective, for example, learning $p(M|D)$ is actually an inference problem. When not all variables are observable, computing point estimates of M needs inference to impute the missing data.

1.1 Likelihood

One of the simplest queries one may ask is likelihood estimation. The likelihood estimation of a probability distribution is to compute the probability of the given evidence, where evidence is an assignment of values to a subset of variables. Likelihood estimation involves marginalization of the other variables. Formally, let e and x denote evidence and the remaining variables, respectively, the likelihood of e is

$$P(e) = \sum_{x_1} \cdots \sum_{x_k} P(x_1, \cdots, x_k, e).$$

From the perspective of computational statistics, calculating this is computationally expensive because the computation involves exploring exponentially many configurations.

1.2 Conditional Probability

Another type of queries is the conditional probability of variables given evidence. The probability of variables X given evidence e is

$$P(X|e) = \frac{P(X,e)}{P(e)} = \frac{P(X,e)}{\sum_x P(X=x,e)}.$$

This is called a posteriori belief in X given evidence e.

As in likelihood estimation, calculating a conditional probability involves marginalization of variables that are not of our interest (i.e., the summation part in the equation). We will be covering efficient algorithms for marginalization later in this class.

Calculating a conditional probability is called in different ways depending on the query nodes. When the query node is a terminal variable in a directed graphical model, the inference process is called prediction. But

y_1	y_2	$P(y_1, y_2)$
0	0	0.35
0	1	0.05
1	0	0.3
1	1	0.3

Table 1: MPA

if the query node is an ancestor of the evidence, the inference process is called diagnosis; e.g., the probability of disease/fault given evidence symptoms. For instance, in the deep belief network, a restricted Boltzmann machine with multiple hidden layers, the hidden layers are estimated given data.

1.3 Most Probable Assignment

We may query the most probable assignment (MPA) for a subset of variables in the domain. Given evidence e, query variables Y, and the other variables Z, the MPA of Y is

$$\mathrm{MPA}(Y|e) = \arg\max_y P(y|e) = \arg\max_y \sum_z P(y, z|e).$$

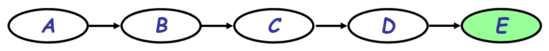
This is the maximum a posteriori configuration of Y.

The MPA of a variable depends on its context. For example, in Table 1, the MPA of Y_1 is 1 because $P(Y_1 = 0) = 0.35 + 0.05 = 0.4$ and $P(Y_1 = 1) = 0.3 + 0.3 = 0.6$. However, when we include Y_2 as context, the MPA of (Y_1, Y_2) is $(0, 0)$ because $P(Y_1 = 0, Y_2 = 0) = 0.35$ is the highest value in the table.

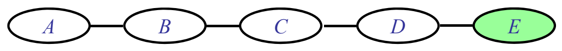
This is related to multi-task learning in machine learning. Multi-task learning indicates solving multiple learning tasks jointly instead of learning task-specific models separately ignoring context. For example, we may improve the accuracy of a classifier of having lunch and that of a classifier of having coffee if we jointly model having lunch and having coffee together, compared to making the two classifiers separately.

2 Approaches to Inference

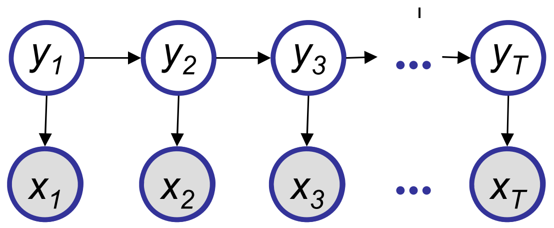
There are two types of inference techniques: exact inference and approximate inference. Exact inference algorithms calculate the exact value of probability $P(X|Y)$. Algorithms in this class include the elimination algorithm, the message-passing algorithm (sum-product, belief propagation), and the junction tree algorithms. We will not cover the junction tree algorithms in the class because they are outdated and confusing. The time complexity of exact inference on arbitrary graphical models is NP-hard. However, we can improve efficiency for particular families of graphical models. Approximate inference techniques include stochastic simulation and sampling methods, Markov chain Monte Carlo methods, and variational algorithms.



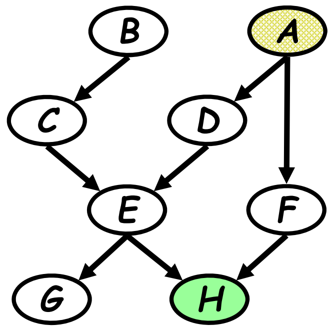
(a) Chain



(b) Conditional Random Field



(c) Hidden Markov Model



(d) General Directed Graphical Model

Figure 1: Graphical Model Examples

3 Elimination

3.1 Chains

Let's first consider inference on the simple chain in Figure 1a. The probability $P(E = e)$ can be calculated as

$$P(e) = \sum_d \sum_c \sum_b \sum_a P(a, b, c, d, e).$$

This naive summation enumerates over exponentially many configurations of the variables and thus is inefficient. But if we use the chain structure, the marginal probability can be calculated as

$$\begin{aligned}
P(e) &= \sum_d \sum_c \sum_b \sum_a P(a)P(b|a)P(c|b)P(d|c)P(e|d) \\
&= \sum_d P(e|d) \sum_c P(d|c) \sum_b P(c|b) \sum_a P(a)P(b|a) \\
&= \sum_d P(e|d) \sum_c P(d|c) \sum_b P(c|b)P(b) \\
&= \sum_d P(e|d) \sum_c P(d|c)P(c) \\
&= \sum_d P(e|d)P(d).
\end{aligned}$$

The summation in each line involves enumerating over only two variables, and there are four such summations. Therefore, in general, the time complexity of the elimination algorithm is $O(nk^2)$, where n is the number of variables and k is the possible values of each variable. That is, for simple chains, exact inference can be done in polynomial time as opposed to the exponential time for the naive approach.

3.2 Hidden Markov Model

Let's consider the hidden Markov model in Figure 1c. The conditional probability of variable Y_i given X can be calculated as

$$\begin{aligned}
P(y_i|x_1,\cdots,x_T) &= \sum_{y_1}\cdots\sum_{y_{i-1}}\sum_{y_{i+1}}\cdots\sum_{y_T} P(y_1,\cdots,y_T,x_1,\cdots,x_T) \\
&= \sum_{y_1}\cdots\sum_{y_{i-1}}\sum_{y_{i+1}}\cdots\sum_{y_T} P(y_1)P(x_1|y_1)\cdots P(y_T|y_{T-1})P(x_T|y_T) \\
&= \sum_{y_2}\cdots\sum_{y_{i-1}}\sum_{y_{i+1}}\cdots\sum_{y_T} P(x_2|y_2)\cdots P(y_T|y_{T-1})P(x_T|y_T)\sum_{y_1}P(y_1)P(x_1|y_1)P(y_2|y_1) \\
&= \sum_{y_2}\cdots\sum_{y_{i-1}}\sum_{y_{i+1}}\cdots\sum_{y_T} P(x_2|y_2)\cdots P(y_T|y_{T-1})P(x_T|y_T)m(x_1,y_2) \\
&= \sum_{y_3}\cdots\sum_{y_{i-1}}\sum_{y_{i+1}}\cdots\sum_{y_T} P(x_3|y_3)\cdots P(y_T|y_{T-1})P(x_T|y_T)m(x_1,x_2,y_3) \\
&= \vdots
\end{aligned}$$

In fact, $m(x_1, y_2)$ is equal to the marginal probability $P(x_1, y_2)$.

3.3 Undirected Chains

Let's consider the undirected chain in Figure 1b. With the elimination algorithm, the marginal probability $P(E = e)$ can be calculated similarly as

$$\begin{aligned}
P(e) &= \sum_d\sum_c\sum_b\sum_a \frac{1}{Z}\phi(b,a)\phi(c,b)\phi(d,c)\phi(e,d) \\
&= \frac{1}{Z}\sum_d\sum_c\sum_b \phi(c,b)\phi(d,c)\phi(e,d)\sum_a\phi(b,a) \\
&= \vdots
\end{aligned}$$

In general, we can view the task at hand as that of computing the value of an expression of the form

$$\sum_z \prod_{\phi\in F}\phi,$$

where F is a set of factors. This task is called the sum-product inference.

4 Variable Elimination

4.1 The Algorithm

We can extend the elimination algorithm to arbitrary graphical models. For a directed graphical model,

$$P(X_1, e) = \sum_{x_n}\cdots\sum_{x_3}\sum_{x_2}\prod_{i\in V}P(x_i|pa_i).$$

For variable elimination, we repeat:

1. Move all irrelevant terms outside of the innermost sum.
2. Perform the innermost sum and get a new term.
3. Insert the new term into the product.

Note that the elimination algorithm has no benefit if the innermost term includes all variables, that is, x_i is dependent on all the other variables. However, in most problems, the number of variables in the innermost term is less than the total number of variables.

For undirected graphical models,

$$P(X_1|e) = \frac{\phi(X_1, e)}{\sum_{x_1} \phi(X_1, e)}.$$

Let's consider the graphical model in Figure 1d. The joint probability distribution factorizes to

$$P(a, b, c, d, e, f, g, h) = P(a)P(b)P(c|b)P(d|a)P(e|c, d)P(f|a)P(g|e)P(h|e, f).$$

To calculate the conditional probability $P(A|h)$, we first choose an elimination order:

$$H, G, F, E, D, C, B.$$

We condition on the evidence node H by fixing its value to h. To treat marginalization and conditioning as formally equivalent, we can define an evidence potential $\delta(h = \tilde{h})$ whose value is one if the inner statement is true and zero otherwise. Then, we obtain

$$P(H = \tilde{h}|e, f) = \sum_h P(h|e, f)\delta(h = \tilde{h}).$$

The conditional probability $P(a|h)$ is calculated as

$$\begin{aligned}
P(a, \tilde{h}) &= \sum_b \sum_c \sum_d \sum_e \sum_f \sum_g P(a)P(b)P(c|b)P(d|a)P(e|c, d)P(f|a)P(g|e)P(h|e, f) \\
&= P(a) \sum_b P(b) \sum_c P(c|b) \sum_d P(d|a) \sum_e P(e|c, d) \sum_f P(f|a) \sum_g P(g|e) \sum_h P(h|e, f)\delta(h = \tilde{h}) \\
&= P(a) \sum_b P(b) \sum_c P(c|b) \sum_d P(d|a) \sum_e P(e|c, d) \sum_f P(f|a) \sum_g P(g|e)m_h(e, f) \\
&= P(a) \sum_b P(b) \sum_c P(c|b) \sum_d P(d|a) \sum_e P(e|c, d) \sum_f P(f|a)m_h(e, f) \sum_g P(g|e) \\
&= P(a) \sum_b P(b) \sum_c P(c|b) \sum_d P(d|a) \sum_e P(e|c, d) \sum_f P(f|a)m_h(e, f) \\
&= P(a) \sum_b P(b) \sum_c P(c|b) \sum_d P(d|a) \sum_e P(e|c, d)m_f(a, e) \\
&= P(a) \sum_b P(b) \sum_c P(c|b) \sum_d P(d|a)m_e(a, c, d) \\
&= P(a) \sum_b P(b) \sum_c P(c|b)m_d(a, c) \\
&= P(a) \sum_b P(b)m_c(a, b, c) \\
&= P(a)m_b(a)
\end{aligned}$$

Therefore,

$$P(a|\tilde{h}) = \frac{P(a, \tilde{h})}{P(\tilde{h})} = \frac{P(a)m_b(a)}{\sum_a P(a)m_b(a)}.$$

4.2 Complexity of Variable Elimination

In one elimination step, we should compute:

$$m_x(y_1, ..., y_k) = \sum_x m'_x(x, y_1, ..., y_k)m'_x(x, y_1, ..., y_k) = \prod_{i=1}^{k} m_i(x, \mathbf{y}_{c_i})$$

In the first equation, we should do $|Val(X)| \cdot \prod_i |Val(\mathbf{y}_{c_i})|$ additions. In the second equation, we should do $k \cdot |Val(X) \cdot \prod_i |Val(\mathbf{y}_{c_i})|$ multiplications. Therefore the computational complexity is *exponential* in number of variables in the intermediate factor.

5 Understanding Variable Elimination

The equations in the above section describe the process of Elimination from the perspective of mathematics. While we can describe the Elimination process from the perspective of graph - *graph elimination algorithm.* There are mainly two steps in the graph elimination algorithm: **moralization** and **(undirected) graph elimination**.

5.1 Moralization

Moralization is a process of converting a directed acyclic graph(DAG) into "equivalent" undirected graph. Its procedure is:

- Starting from an input DAG
- Connect nodes if they share a common child.
- Make directed edges to undirected edges.

5.2 Graph Elimination

The graph elimination algorithm is:

```
**Input** undirected GM or moralized DAG & the elimination order $I$
**for** each node $X_i$ in $I$
- connect all of the remaining neighbors of $X_i$
- remove $X_i$ from the graph
**end**
```

5.3 Graph Elimination and Marginalization

Now we can interpret the Elimination algorithm from the perspective of graph elimination algorithm. As shown in the Figure 2 below, the *summation* step in Elimination can be represented by an *elimination* step in graph elimination algorithm. In addition, the *intermediate terms* in Elimination correspond to the *elimination cliques* resulted from graph elimination algorithm (Figure 3a). In addition, we can also construct a *clique tree* to represent the elimination process (Figure 3b).

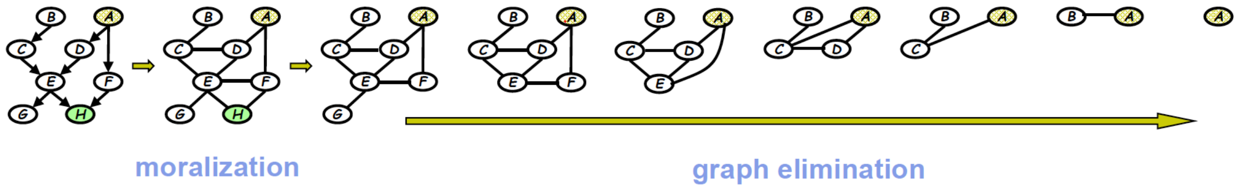
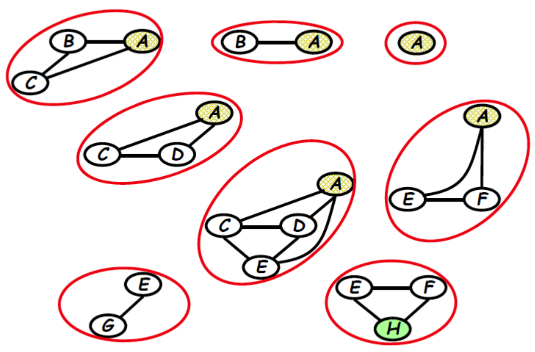
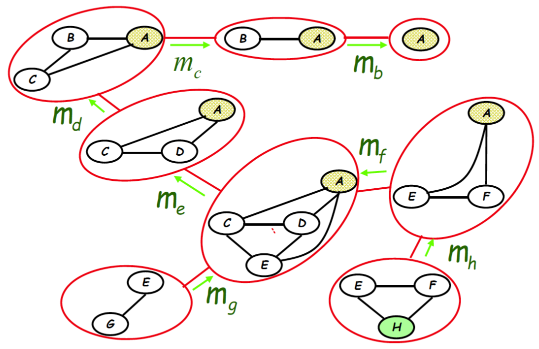


Figure 2: A graph elimination

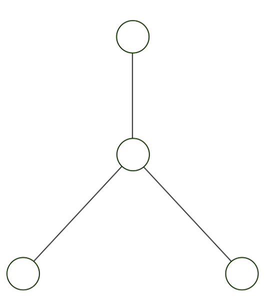


(a) Elimination cliques

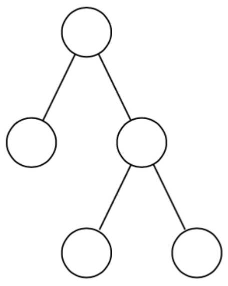


(b) Clique tree

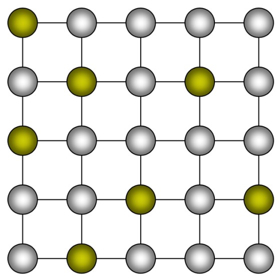
Figure 3: Cliques



(a) Star



(b) Tree



(c) Ising Model

Figure 4: Tree-Width Examples

5.4 Complexity

The overall complexity is determined by the number of the largest elimination clique. A "good" elimination will make the largest clique relatively small. *Tree-width* k is introduced to study this problem. Its definition is one less than the smallest achievable value of the cardinality of the largest elimination clique, ranging over all possible elimination ordering. However, finding such k as well as the "best" elimination ordering is NP-hard. For some of the graphs such as stars (Figure 4a) ($k = 2 - 1$) and trees (Figure 4b) ($k = 2 - 1$), we can easily get their tree-width k, while for graphs like ising model (Figure 4c), it is very hard to compute the k.

6 Message Passing

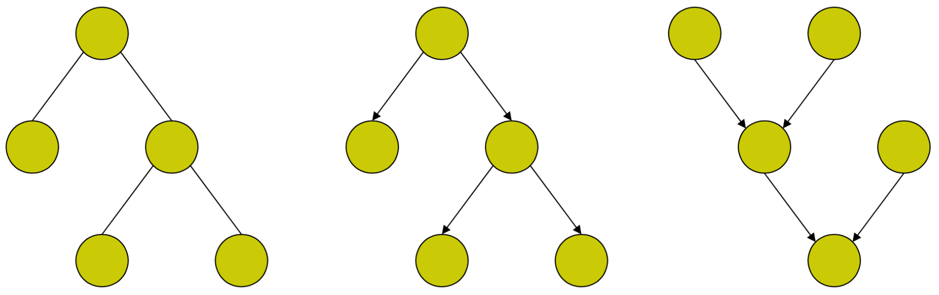


Figure 5: Tree GMs: from left to right *undirected tree*, *directed tree*, *polytree*

6.1 From Elimination to Message Passing

One of the limitation of the elimination is that it only answers one query (e.g., on one node). To answer other queries (nodes) as well, the notion of *message* is introduced. Each step of elimination is actually a message passing on a clique tree. Although different query has different clique tree, the message passing through the tree can be reused.
There are mainly three types of trees (Figure 5): **undirected tree**, **directed tree** and **polytree**. In fact, directed and undirected trees are equivalent, since:
(1) Undirected trees can be converted to directed by choosing a root and directing all edges away from it.
(2) A directed tree and the corresponding undirected tree make the same conditional independence assertions.
(3) Parameterizations are essentially the same. $p(x) = \frac{1}{Z}(\prod_{i\in V}\psi(x_i)\prod_{(i,j)\in E}\psi(x_i, x_j))$

6.2 Elimination on A Tree

We can show that elimination on trees is equivalent to message passing along tree branches. As shown in figure 6a, let $m_{ji}(x_i)$ denote the factor resulting from eliminating variables from bellow up to i, which is a function of x_i:

$$m_{ji}(x_i) = \sum_{x_j}(\psi(x_j)\psi(x_i, x_j)\prod_{k\in N(j)\setminus i} m_{kj}(x_j))$$

This is equivalent to the message from j to i.

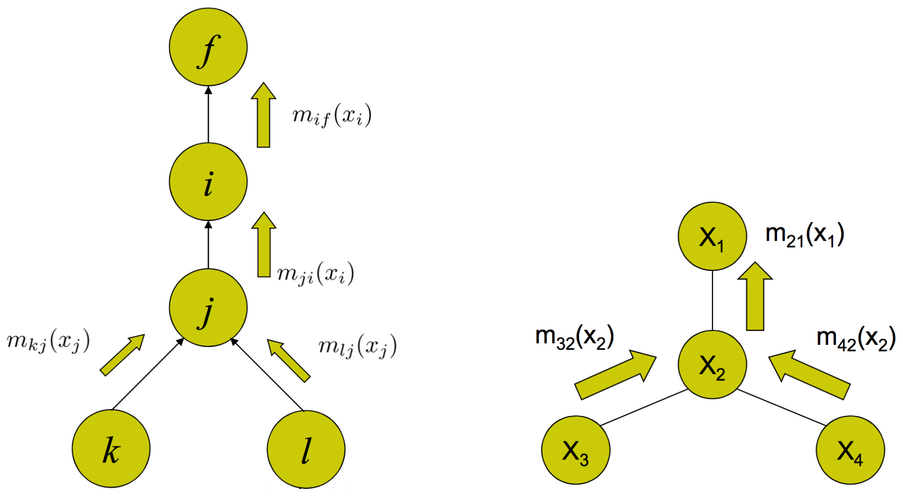
The process of *elimination* or *message passing* can be described as:

- Choose query node f as the root of the tree.
- View tree as a directed tree with edges pointing towards leaves from f.
- Elimination ordering based on depth-first traversal.
- Elimination of each node can be considered as *message-passing* (or *Belief Propagation*) directly along tree branches, rather than on some transformed graphs.

Therefore, we can use the tree itself as a data-structure to do general inference.

6.3 The Message Passing Protocol

To efficiently compute the marginal distributions of all nodes in a graph, a *Message Passing Protocol* (Figure 6b) is introduced: *A node can send a message to its neighbors when (and only when) it has received messages from all its other neighbors.* Based on this protocol, a naive computing approach is considering each node as the root, and executing the message passing algorithm for it. The complexity of the naive approach is NC, where N is the number of nodes and C is the complexity of a complete message passing.



(a) Message Passing (b) Message Passing Protocol

Figure 6: Message Passing