Lecture 3: Exact inference & Message Passing

Thibault Randrianarisoa

University of Toronto, Winter 2026

January 20, 2026

Table of contents

Exact Inference

Inference as Conditional Distribution

- We explore inference in probabilistic graphical models (PGMs):
  - x_E = The observed evidence
  - x_F = The unobserved variable we want to infer
  - $x_R = x \backslash \{x_F, x_E\}$ = Remaining variables, extraneous to query

- For inference, focus on computing the conditional probability distribution:

$$p(x_F|x_E) = \frac{p(x_F, x_E)}{p(x_E)} = \frac{p(x_F, x_E)}{\sum_{x_F} p(x_F, x_E)}$$

- for which, we marginalize out these extraneous variables, focusing on the joint distribution over evidence and subject of inference:

$$p(x_F, x_E) = \sum_{x_R} p(x_F, x_E, x_R)$$

Spoiler

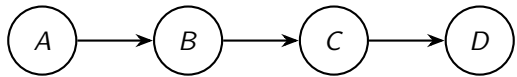
Order in which we marginalize affects the computational cost!

Our main tool is variable elimination, which means marginalizing out one variable at a time:

- A simple and general exact inference algorithm in any probabilistic graphical model.
- Computational complexity depends on the graph structure.
- Dynamic programming avoids enumerating all variable assignments.

Example: Simple Chain

Lets start with the example of a simple chain:



We want to compute $p(D)$ with no evidence variables.

- Variables: $x_F = \{D\}$, $x_E = \{\}$, $x_R = \{A, B, C\}$.
- Factorization: $p(A, B, C, D) = p(A)p(B|A)p(C|B)p(D|C)$.
- Assume each variable can take on k different values.

Example: Simple Chain

The goal is to compute the marginal $p(D) = \sum_{A,B,C} p(A, B, C, D)$ for any possible value of D.

- **Naive approach:** Summing over all assignments has exponential cost $O(k^{n=4})$.

$$\begin{aligned} p(D) &= \sum_{A,B,C} p(A, B, C, D) \\ &= \sum_{C} \sum_{B} \sum_{A} p(A)p(B|A)p(C|B)p(D|C) \end{aligned}$$

- **Variable Elimination:** Choose an elimination ordering:

$$p(D) = \sum_{C} p(D|C) \left(\sum_{B} p(C|B) \left(\sum_{A} p(A)p(B|A) \right) \right)$$

Example: Simple Chain

- This reduces the complexity by first computing terms that appear across the other sums.

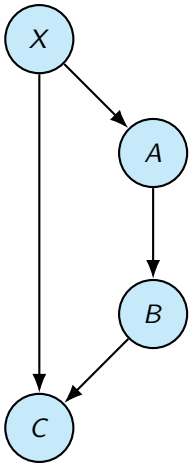
$$
\begin{aligned}
p(D) &= \sum_C p(D|C) \sum_B p(C|B) \sum_A p(A)p(B|A) \\
&= \sum_C p(D|C) \sum_B p(C|B)p(B) \\
&= \sum_C p(D|C)p(C)
\end{aligned}
$$

- e.g. for each value of B, we have to compute the sum $\sum_A p(A)p(B|A)$, which takes $k \cdot k = k^2$ operations.
- The cost of performing inference on the chain in this manner is $\mathcal{O}(nk^2)$. In comparison, generating the full joint distribution and marginalizing over it has complexity $\mathcal{O}(k^n)$!.

Best Elimination Ordering

- The complexity of variable elimination depends heavily on the ordering!
- Unfortunately, finding the best elimination ordering is **NP-hard**.
- However, in chains and trees, the best elimination ordering is sometimes clear.
- **Intuition:**
  - Marginalizing over nodes with no children can be done first.
  - Start with nodes that come early in the induced ordering of the DAG.

Intermediate Factors



- In general, eliminating does not produce a valid marginal or conditional distribution of the graphical model.
- In the example on the left, the distribution is given by:

$$p(X, A, B, C) = p(X)p(A|X)p(B|A)p(C|B, X)$$

Marginalizing over X gives

$$\begin{aligned} p(A, B, C) &= \sum_X p(X)p(A|X)p(B|A)p(C|B, X) \\ &= p(B|A) \sum_X p(X)p(A|X)p(C|B, X) \end{aligned}$$

- The resulting term $\sum_X p(X)p(A|X)p(C|B, X)$ does not correspond to a valid conditional or marginal distribution because it is unnormalized.

Intermediate Factors

- We introduce **factors** ϕ which are not necessarily normalized distributions, but which describe the local relationship between random variables.
- In the above example:

$$\begin{aligned} p(A,B,C) &= \sum_X p(X)p(A|X)p(B|A)p(C|B,X) \\ &= \sum_X \phi_1(X)\phi_2(A,X)\phi_3(A,B)\phi_4(X,B,C) \\ &= \phi_3(A,B)\sum_X \phi_1(X)\phi_2(A,X)\phi_4(X,B,C) \\ &= \phi_3(A,B)\tau(A,B,C) \end{aligned}$$

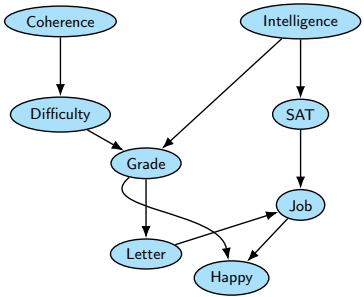
- Marginalizing over X we introduce a new factor, denoted by τ.

Abstractly, computing $p(x_F|x_E)$ is given by the **sum-product** algorithm:

$$p(x_F|x_E) \propto \tau(x_F, x_E) = \sum_{x_R} \prod_{C \in \mathcal{F}} \phi_C(x_C)$$

- For DAGs: $\mathcal{F}$ is given by sets of the form $\{i\} \cup \text{parents}(i)$.

Example



- This describes a factorization:

$$p(C,D,I,G,S,L,H,J) = p(C)p(D|C)p(I) \times p(G|D,I)p(L|G)p(S|I) \times p(J|S,L)p(H|J,G)$$

For notational convenience, we write the conditionals as factors.

$$\Phi = \{\phi(C), \phi(C,D), \phi(I), \phi(G,D,I), \phi(L,G), \phi(S,I), \phi(J,S,L), \phi(H,J,G)\}$$

If we are interested in inferring the marginal probability of getting a job, $P(J)$ we can do exact inference on the joint distribution by marginalizing according to a specific variable elimination ordering.

Example

Elimination Ordering $\prec \{C, D, I, H, G, S, L\}$

$$
\begin{aligned}
p(J) &= \sum_L \sum_S \phi(J, L, S) \sum_G \phi(L, G) \sum_H \phi(H, G, J) \sum_I \phi(S, I)\phi(I) \sum_D \phi(G, D, I) \underbrace{\sum_C \phi(C)\phi(C, D)}_{\tau(D)} \\
&= \sum_L \sum_S \phi(J, L, S) \sum_G \phi(L, G) \sum_H \phi(H, G, J) \sum_I \phi(S, I)\phi(I) \underbrace{\sum_D \phi(G, D, I)\tau(D)}_{\tau(G,I)} \\
&= \sum_L \sum_S \phi(J, L, S) \sum_G \phi(L, G) \sum_H \phi(H, G, J) \underbrace{\sum_I \phi(S, I)\phi(I)\tau(G, I)}_{\tau(S,G)} \\
&= \sum_L \sum_S \phi(J, L, S) \sum_G \phi(L, G)\tau(S, G) \underbrace{\sum_H \phi(H, G, J)}_{\tau(G,J)} \\
&= \sum_L \sum_S \phi(J, L, S) \underbrace{\sum_G \phi(L, G)\tau(S, G)\tau(G, J)}_{\tau(J,L,S)} \\
&= \sum_L \underbrace{\sum_S \phi(J, L, S)\tau(J, L, S)}_{\tau(J,L)} \\
&= \underbrace{\sum_L \tau(J, L)}_{\tau(J)} \\
&= \tau(J)
\end{aligned}
$$

Complexity of Variable Elimination Ordering

Complexity is determined by the number of variables inside each sum:

$$O(m \cdot k^{N_{max}})$$

- m: Number of initial factors.
- k: Number of states each random variable takes (assumed to be equal here).
- N_i: Number of random variables inside each sum $\sum_i$.
- $N_{max} = \max_i N_i$: The number of variables inside the largest sum.

As discussed previously, variable elimination ordering determines the computational complexity.

Elimination Ordering $\prec \{C, D, I, H, G, S, L\}$

- Here are all the initial factors:

$$\mathcal{F} = \{\phi(C), \phi(C, D), \phi(I), \phi(G, D, I), \phi(L, G), \phi(S, I), \phi(J, S, L), \phi(H, J, G)\}$$
$$\implies m = |\mathcal{F}| = 8$$

- Here are the sums, and the number of variables that appear in them

$$\underbrace{\sum_{C} \phi(C)\phi(C, D)}_{N_C=2} \quad \underbrace{\sum_{D} \phi(G, D, I)\tau(D)}_{N_D=3} \quad \underbrace{\sum_{I} \phi(S, I)\phi(I)\tau(G, I)}_{N_I=3}$$

$$\underbrace{\sum_{H} \phi(H, G, J)}_{N_H=3} \quad \underbrace{\sum_{G} \phi(L, G)\tau(S, G)\tau(G, J)}_{N_G=4} \quad \underbrace{\sum_{S} \phi(J, L, S)\tau(J, L, S)}_{N_S=3}$$

$$\underbrace{\sum_{L} \tau(J, L)}_{N_L=2} \implies \text{the largest sum is } N_G = 4.$$

- For simplicity, assume all variables take on k states. So the complexity of the variable elimination under this ordering is $O(8 * k^4)$.

Summary

Variable elimination:

- Used for exact inference in PGMs.
- Ordering can significantly reduce computational complexity.
- Overall complexity is $O(m \cdot k^{N_{max}})$ and can be computed once an ordering is chosen.

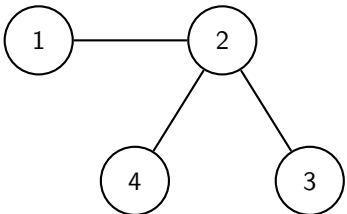
Message passing

Variable Elimination Order and Trees

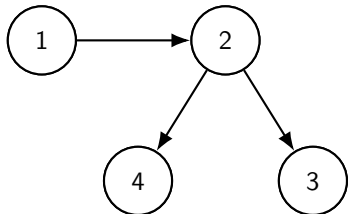
- **First part:** we can do exact inference by variable elimination: i.e. to compute $p(A|C)$, we can marginalize $p(A, B|C)$ over every variable in B, one at a time.
- Computational cost is determined by the graph structure, and the elimination ordering.
- Determining the optimal elimination ordering is hard.
- Even if we do, the resulting marginalization might also be unreasonably costly.
- Fortunately, for **trees**, any elimination ordering that goes from the leaves inwards towards any root will be optimal.
- You can think of trees as just chains which sometimes branch.

Directed Trees as Undirected Models

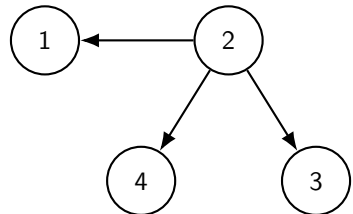
- A **directed tree** is a DAG where every node has at most one parent.
- We can actually ignore the arrows and treat it as an undirected tree.



(a) Undirected



(b) Directed (Root 1)



(c) Directed (Root 2)

Joint Distribution Parameterization

- Consider the factorization for graph (b) (Root at 1):

$$p^*(\mathbf{z}) \propto p^*(z_1)p^*(z_2|z_1)p^*(z_3|z_2)p^*(z_4|z_2)$$

- We can rewrite this solely in terms of marginals (nodes) and pairwise joints (edges):

$$\begin{aligned} p^*(\mathbf{z}) &= p^*(z_1)\frac{p^*(z_1,z_2)}{p^*(z_1)}\frac{p^*(z_2,z_3)}{p^*(z_2)}\frac{p^*(z_2,z_4)}{p^*(z_2)} \\ &= p^*(z_1)p^*(z_2)p^*(z_3)p^*(z_4)\frac{p^*(z_1,z_2)}{p^*(z_1)p^*(z_2)}\frac{p^*(z_2,z_3)}{p^*(z_2)p^*(z_3)}\frac{p^*(z_2,z_4)}{p^*(z_2)p^*(z_4)} \end{aligned}$$

- Notice the direction of the arrows has disappeared! This depends only on the cliques (edges) and nodes.
- We obtain the same with graph (c)

Symmetric Factorization for Trees

- We can generalize this result. For any tree structure, the joint distribution can be written as a product of local marginals divided by "separators" (node marginals).

Tree Factorization with Potentials

$$p^*(\mathbf{z}) \propto \prod_{s \in \mathcal{V}} p^*(z_s) \prod_{(s,t) \in \mathcal{E}} \frac{p^*(z_s, z_t)}{p^*(z_s) p^*(z_t)} = \prod_{s \in \mathcal{V}} p^*(z_s) \prod_{(s,t) \in \mathcal{E}} \psi_{st}(z_s, z_t)$$

- This form treats every edge (s, t) as a symmetric potential:

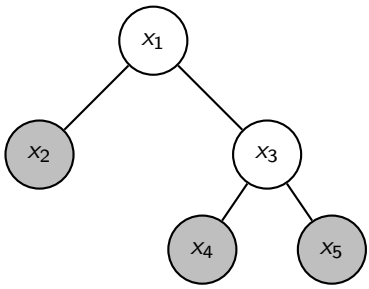
$$\psi_{st}(z_s, z_t) = \frac{p^*(z_s, z_t)}{p^*(z_s) p^*(z_t)}$$

- This confirms that for trees, the directed and undirected views are equivalent.

Message Passing (Belief Propagation)

- **Problem:** Variable Elimination finds $p(x_F|x_E)$ for a specific (set of) query node x_F.
- If we want the marginals for **every** node in the graph, running VE multiple times is inefficient.
- **Solution:** Message Passing (Sum-Product Algorithm on tree).
- We view the graph edges as communication channels, sharing information.

Example: Inference in Trees



- The joint distribution is $p(x) = \frac{1}{Z} \prod_{i \in \mathcal{V}} \psi_i(x_i) \prod_{(i,j) \in \mathcal{E}} \psi_{ij}(x_i, x_j)$.
- Want to compute $p(x_3|x_E)$,

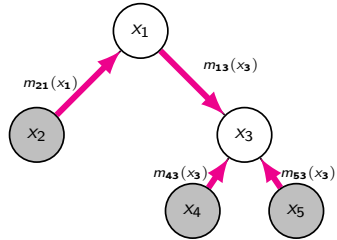
$$x_E = (\bar{x}_2, \bar{x}_4, \bar{x}_5), \quad x_R = x_1$$

- We have $p(x_3|x_E) \propto p(x_3, x_E)$ (meaning that $p(x_3|x_E) = \frac{p(x_3, x_E)}{\sum_{x'_3} p(x'_3, x_E)}$ and $Z^E = \sum_{x'_3} p(x'_3, x_E)$)

$$p(x_3|x_E) = \frac{1}{Z^E} \sum_{x_1} \psi_1(x_1)\psi_2(\bar{x}_2)\psi_3(x_3)\psi_4(\bar{x}_4)\psi_5(\bar{x}_5)\psi_{12}(x_1, \bar{x}_2)\psi_{13}(x_1, x_3)\psi_{34}(x_3, \bar{x}_4)\psi_{35}(x_3, \bar{x}_5)$$

We write the variable elimination algorithm revealing additional structure.

Example: Inference in Trees



$$p(x_3|x_E) = \frac{1}{Z^E}\sum_{x_1}\psi_1(x_1)\psi_2(\bar{x}_2)\psi_3(x_3)\psi_4(\bar{x}_4)\psi_5(\bar{x}_5)\psi_{12}(x_1,\bar{x}_2)\psi_{13}(x_1,x_3)\psi_{34}(x_3,\bar{x}_4)\psi_{35}(x_3,\bar{x}_5)$$

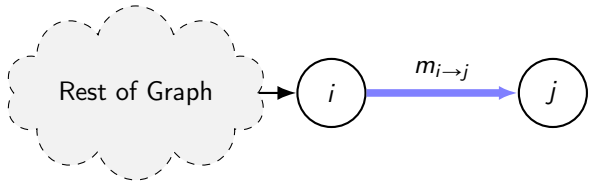
$$= \frac{1}{Z^E}\underbrace{\psi_4(\bar{x}_4)\psi_{34}(x_3,\bar{x}_4)}_{m_{43}(x_3)}\underbrace{\psi_5(\bar{x}_5)\psi_{35}(x_3,\bar{x}_5)}_{m_{53}(x_3)}\psi_3(x_3)\sum_{x_1}\psi_1(x_1)\psi_{13}(x_1,x_3)\underbrace{\psi_2(\bar{x}_2)\psi_{12}(x_1,\bar{x}_2)}_{m_{21}(x_1)}$$

$$= \frac{1}{Z^E}m_{43}(x_3)m_{53}(x_3)\psi_3(x_3)\underbrace{\sum_{x_1}\psi_1(x_1)\psi_{13}(x_1,x_3)m_{21}(x_1)}_{m_{13}(x_3)}$$

$$= \frac{1}{Z^E}\psi_3(x_3)m_{43}(x_3)m_{53}(x_3)m_{13}(x_3) = \frac{\psi_3(x_3)m_{43}(x_3)m_{53}(x_3)m_{13}(x_3)}{\sum_{x_3'}\psi_3(x_3')m_{43}(x_3')m_{53}(x_3')m_{13}(x_3')}$$

What is a "Message"?

- A message $m_{i \to j}(x_j)$ is a **factor** sent from node i to neighbor j.
- **Intuition:** It summarizes the entire branch of the graph "behind" node i, so node j doesn't need to look there.
- It is exactly the intermediate factor τ from Variable Elimination!



Message Passing on Trees

We perform variable elimination from leaves to root, which is the sum product algorithm to compute all marginals. Belief propagation is a message-passing between neighboring vertices of the graph.

- The message sent from variable j to $i \in N(j)$ is

$$m_{j\to i}(x_i) = \sum_{x_j} \psi_j(x_j)\psi_{ij}(x_i, x_j) \prod_{k\in N(j)\setminus i} m_{k\to j}(x_j)$$

  - If x_j is observed, the message is

$$m_{j\to i}(x_i) = \psi_j(\bar{x}_j)\psi_{ij}(x_i, \bar{x}_j) \prod_{k\in N(j)\setminus i} m_{k\to j}(\bar{x}_j)$$

- Once the message passing stage is complete, we can compute our beliefs as

$$b(x_i) \propto \psi_i(x_i) \prod_{j\in N(i)} m_{j\to i}(x_i).$$

- Once normalized, beliefs are the marginals we want to compute!

Belief Propagation Algorithm on Trees

Step 1 Choose root r arbitrarily

Step 2 Pass messages from leafs to r

Step 3 Pass messages from r to leafs

These two passes are sufficient on trees!
$\forall (i,j)$ compute $m_{i\to j}(x_j)$ and $m_{j\to i}(x_i)$.

Step 4 Compute beliefs (marginals)

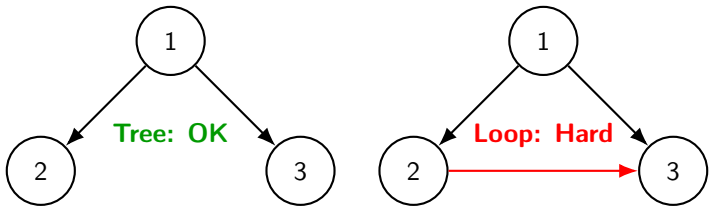
$$\forall i,\ b(x_i) = p(x_i|x_E) \propto \psi_i(x_i) \prod_{j\in N(i)} m_{j\to i}(x_i)$$

One can compute them in two steps:

- Compute unnormalized beliefs $\tilde{b}(x_i) = \psi_i(x_i) \prod_{j\in N(i)} m_{j\to i}(x_i)$
- Normalize them $b(x_i) = \tilde{b}(x_i) / \sum_{x_i'} \tilde{b}(x_i')$.

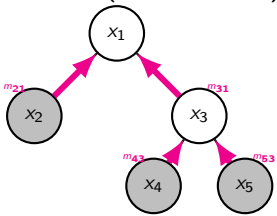
Tree constraint

- Exact Message Passing works simply on graphs without loops (**Singly Connected**).
- Between any two nodes, there is exactly one path.

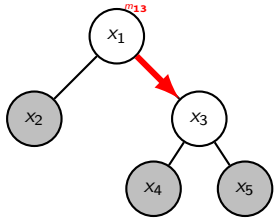


Inference in Trees: Compute $p(x_3|\bar{x}_2, \bar{x}_4, \bar{x}_5)$ and $p(x_1|\bar{x}_2, \bar{x}_4, \bar{x}_5)$

1. Collect (Leaves → Root)



2. Distribute (Root → Leaves)



$$m_{j\to i}(x_i) = \sum_{x_j} \psi_j(x_j)\psi_{ij}(x_i, x_j) \prod_{k\in N(j)\setminus i} m_{k\to j}(x_j)$$

$$b(x_i) \propto \psi_i(x_i) \prod_{j\in N(i)} m_{j\to i}(x_i).$$

- $m_{5\to 3}(x_3) = \psi_5(\bar{x}_5)\psi_{35}(x_3, \bar{x}_5)$
- $m_{2\to 1}(x_1) = \psi_2(\bar{x}_2)\psi_{12}(x_1, \bar{x}_2)$ x_2, x_4, x_5 are observed
- $m_{4\to 3}(x_3) = \psi_4(\bar{x}_4)\psi_{34}(x_3, \bar{x}_4)$
- $m_{3\to 1}(x_1) = \sum_{x_3} \psi_3(x_3)\psi_{13}(x_1, x_3) m_{4\to 3}(x_3) m_{5\to 3}(x_3)$
- $m_{1\to 3}(x_3) = \sum_{x_1} \psi_1(x_1)\psi_{13}(x_1, x_3) m_{2\to 1}(x_1)$
- $b(x_1) \propto \psi_1(x_1) m_{2\to 1}(x_1) m_{3\to 1}(x_1)$
- $b(x_3) \propto \psi_3(x_3) m_{1\to 3}(x_3) m_{4\to 3}(x_3) m_{5\to 3}(x_3)$

- MAP inference: Suppose that instead of marginalizing out x_R we are interested in the most likely configuration $\hat{x} = \arg\max p(x|x_E)$.

- For MAP inference, we maximize over x_j instead of summing over them. This is called **max-product BP** with updates

$$m_{j\to i}(x_i) = \max_{x_j} \psi_j(x_j)\psi_{ij}(x_i, x_j) \prod_{k\in N(j)\setminus i} m_{k\to j}(x_j)$$

- The beliefs are **max-marginals**

$$\hat{b}(x_i) = \max_{x_{\setminus i}} p(x_i, x_{\setminus i}|x_E) \propto \psi_i(x_i) \prod_{j\in N(i)} m_{j\to i}(x_i).$$

- MAP inference: take $\hat{x}_i := \arg\max_{x_i} \hat{b}(x_i)$ for all $i \notin E$.

Summary: Message Passing vs VE

Variable Elimination

- Single Query.
- One-way flow.
- Discards intermediates.

Message Passing

- All-Nodes Query.
- Two-way flow (Collect/Distribute).
- Caches intermediates (messages).

The algorithm we learned is called **sum-product BP**. If we are interested in MAP inference, we can maximize over x_j instead of summing over them. This is called **max-product BP**.