Inference in Graphical Models

Generative and Graphical Models AI60201, Module 2

Adway Mitra

Indian Institute of Technology Kharagpur

5 September 2022

Contents

1. Background
2. Exact Inference
3. Approximate Inference
   - Markov Chain Monte Carlo
   - Variational Inference

Background

Introduction

- A graphical model encodes a joint distribution over a set of random variables, like $p(A, B, C)$
- Can we use it to compute marginal distributions like $p(B)$ or $p(A, C)$?
- Standard approach: sum/integrate out all the remaining variables
- Problems:
  - scales exponentially with number of variables
  - polynomially with number of possible values of each variables (in case of discrete)
  - Integration may not be analytically tractable (in case of continuous)
- Using the graph structure may reduce computational requirements!

Exact and Approximate Inference

- Exact inference: calculate the complete marginal distribution of target variable set
- Possible if all random variables are discrete
- May be infeasible if too many variables involved with complex coupling
- In case of continuous random variables, marginalization by integration may not be tractable
- Approach: *approximate inference*
  - Use sampling-based approach to construct the target distribution rather than calculate
  - Substitute it with another distribution that resembles the original but is easier to manage

Exact Inference

Discrete Marginalization

- Consider a chain graph of four binary variables $A->B->C->D$
- Factorization: $p(A,B,C,D) = p(A) * p(B/A) * p(C/B) * p(D/C)$
- Aim: to find marginalize $p(B)$
- Brute force: $p(B) = \sum_A \sum_C \sum_D p(A,B,C,D)$ (sum up $2^3 = 8$ values, with 3 multiplications each time - total 8*3+7=31 operations)
- Notice: D is present in only one factor $p(D/C)$ (pendent vertex)
- $\sum_D p(D/C) = 1$, D eliminated without residue
- Next $\mu_B^{CD} = \sum_C p(C/B)$ (2 summations) - C eliminated
- Next $\mu_B^A = \sum_A p(A) * p(B/A)$ (4 multiplications, 2 summations)
- Finally $p(B) = \mu_B^A * \mu_B^{CD}$ (2 multiplications)
- Magic of the distributive law!

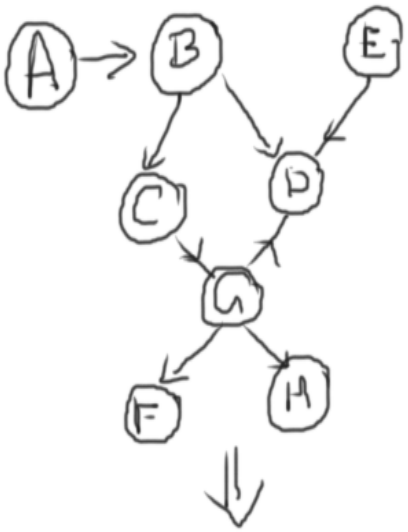
Variable Elimination Algorithm

- The graphical model and its associated factorization
- Input: the target variable set (whose marginal/joint distribution is required)
- Step 1: specify an order to eliminate the remaining
- Set ϕ as the list of factors
- Repeat for each elimination variable in order
  - Select all factors ϕ' from ϕ that involve the current elimination variable
  - Multiply them and sum over all possible values of the elimination variable
  - Result: a function μ (residual) of the other variables covered by the factors in ϕ'
  - Remove the factors ϕ' from ϕ and insert μ into ϕ
- Multiply the remaining factors in ϕ to get the desired distribution

Elimination Order and Residual Graph

- The number of computations in each step of elimination depends on the number of variables covered by the participating factors
- Related to the clique size of the *residual graph*
- Elimination graph: initially the undirected representation of the original graphical model
- To eliminate each variable, remove the corresponding node and its attached edges from the graph
- But add *elimination/residual edges* between the variables of the residual (if not already present)
- Elimination order may be decided at run-time, to keep down the clique size of the elimination graph

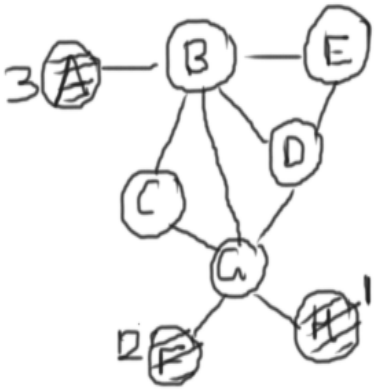
Variable Elimination Example



$$P = p(A) \times p(B/A) \times p(C/B) \times p(E) \times p(G/C) \times p(D/B,E,G) \times p(F/G) \times p(H/G)$$

Elim 1: H $\quad \sum_H p(H/G) = 1$

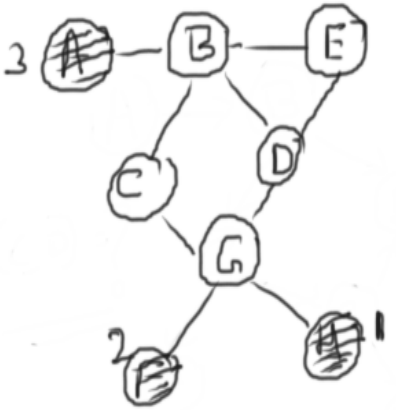
Elim 2: F $\quad \sum_F p(F/G) = 1$



Elim 3: A $\quad \sum_A p(A)\, p(B/A) = N_B$

$$\sum_{H,F,A} P = N_B \times p(C/B) \times p(E) \times p(G/C) \times p(D/B,E,G)$$

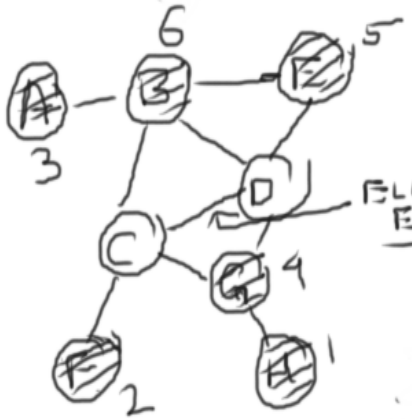
Variable Elimination Example



$$\sum_{H,F,A} p = N_B \times p(C/B) \times p(E) \times p(D/B,E,G) \times p(G/C)$$

Elim 4 : G

$$\sum_{G} p(D/BEG)\, p(G/C) = N_{BCDE}$$



Elim 5 : E

$$\sum_{E} p(E) \times N_{BCDE} = N_{BCD}$$

Elim 6 : B

$$\sum_{B} N_B \times p(C/B) \times N_{BCD} = N_{CD}$$

[FINAL ANSWER]

Another view of Variable Elimination

- Each elimination variable sends a *message* to its neighbors
- The message encodes its own information and the information it had received earlier from other nodes
- Easiest to understand in the chain graph over $X_1, \ldots, X_N$, to compute $p(X_n)$
- Message from the left: $\mu_2^1 = \sum_{X_1} \psi(X_1, X_2)$ (from node 1 to node 2)
- Propagates to the right: $\mu_3^2 = \sum_{X_2} \psi(X_2, X_3)\mu_2^1$ (from node 2 to node 3)
- Till it reaches target (node n): $\mu_n^{n-1} = \sum_{X_{(n-1)}} \psi(X_{(n-1)}, X_n)\mu_{(n-1)}^{(n-2)}$
- Similar messages from the right: $\mu_{N-1}^N = \sum_{X_N} \psi(X_{N-1}, X_N)$ (from node N to node $N-1$) Propagate to the left: $\mu_{N-2}^{N-1} = \sum_{X_{N-1}} \psi(X_{N-1}, X_{N-2})\mu_{N-1}^N$ (from node $N-1$ to node N)
- Till it reaches the target (node n): $\mu_n^{n+1} = \sum_{X_{(n+1)}} \psi(X_{(n+1)}, X_n)\mu_{(n+1)}^{(n+2)}$
- And finally, $p(X_n) = \mu_n^{n-1} * \mu_n^{n+1}$

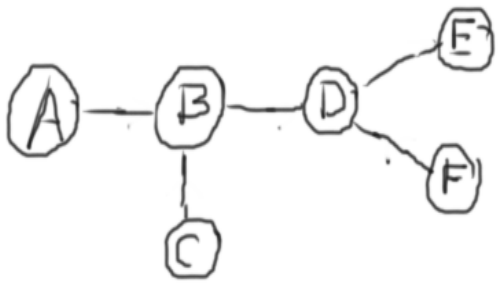
Message Passing

- Consider a situation where you need marginal distributions of all the variables
- Can be done by running Variable Elimination for all nodes
- Many factors have to be calculated repeatedly
- Idea: each node can calculate the factors involving their neighbors locally, store them and also pass these factors (messages) to their neighbors for computing further factors
- Pendant vertices should initiate process
- Each vertex u with k neighbors can wait to receive messages from $(k-1)$ of them. Then it combines those and sends to the remaining neighbor v
- When u receives message back from v, it multiplies all messages together locally, and transmits the result back to remaining nodes
- Whole process of passing messages to continue till each vertix has received messages from all others

Message Passing in General Graphs

- Message Passing is also called *Belief Propagation* or *sum-product*
- Message Passing works on trees (graphs without cycles)
- It can work on both directed and undirected graphical models
- Directed graphical models can be expressed as factor graphs directly
- In case of graphs with cycles, message-passing gets into trouble
- Solution: Loopy Belief Propagation
- *Junction Tree Algorithm*: in case of chordal graphs (where each cycle of length 4 or more is triangulated, i.e. has a sub-cycle of size 3)

Message Passing Demo



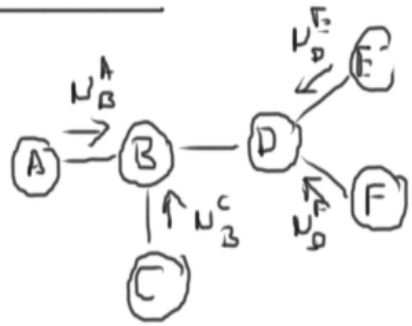
$$P \propto \psi_{AB} \times \psi_{BC} \times \psi_{BD} \times \psi_{DE} \times \psi_{DF}$$

Messages M_j^i : from i to j

B receives:

$$M_B^A = \sum_A \psi_{AB}$$

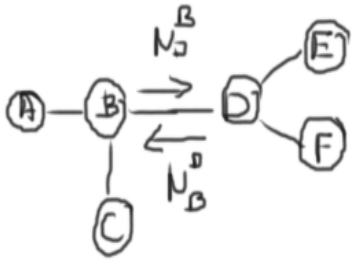
$$M_B^C = \sum_C \psi_{BC}$$



D receives:

$$M_D^E = \sum_E \psi_{DE}$$

$$M_D^F = \sum_F \psi_{DF}$$



$$p(B) \propto M_B^A \times M_B^C \times M_B^D$$

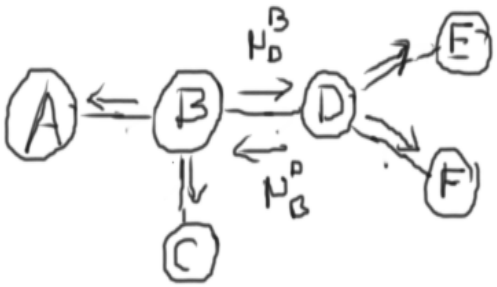
$$p(D) \propto M_D^E \times M_D^F \times M_D^B$$

Next

$$M_D^B = \sum_B M_B^A M_B^C \psi_{BD}$$

$$M_B^D = \sum_D M_D^F M_D^E \psi_{DB}$$

Factor Graph for Message Passing



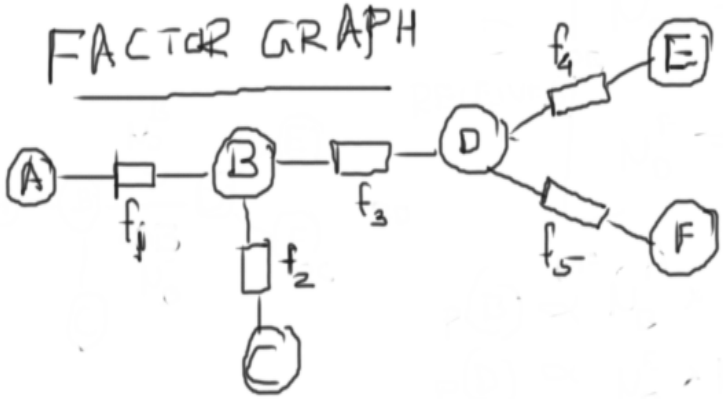
$$\mu_A^B = \sum_B \psi_{AB}\, \mu_B^D \qquad p(A) \propto \mu_A^B$$

$$\mu_C^B = \sum_B \psi_{BC}\, \mu_C^B \qquad p(C) \propto \mu_C^B$$

$$\mu_E^D = \sum_D \psi_{DE}\, \mu_D^B \qquad p(E) \propto \mu_E^D$$

$$\mu_F^D = \sum_D \psi_{DF}\, \mu_D^B \qquad p(F) \propto \mu_F^D$$

FACTOR GRAPH



$$\mu_{A\to f_1} = \psi_{AB} \qquad \mu_{C\to f_2} = \psi_{BC}$$

$$\mu_{f_1\to B} = \sum_A \mu_{A\to f_1} \qquad \mu_{f_2\to B} = \sum_C \mu_{C\to f_2}$$

$$\mu_{B\to f_3} = \mu_{f_1\to B} * \mu_{f_2\to B}$$

$$\mu_{f_3\to D} = \sum_B \mu_{B\to f_3}$$

Factor Graph

- Factor graph: undirected bipartite graph with **Factor nodes** corresponding to each factor of distribution (unary, binary, multi-variable)
- Undirected edges between every variable node and node of those factors which involve the variable
- Designate one node as ROOT, pendant nodes are leaves
- Each leaf-factor sends message 1 to its attached variable-node $\mu_{f->x} = 1$
- Each variable-node that has one factor-neighbor, sends a message equal to the factor $\mu_{x->f} = \psi(f)(x)$
- Each variable-node with multiple factor-neighbors, multiplies messages from all-but-one neighbors, and sends that product to remaining factor-neighbor $\mu_{x->f} = \prod_{f':NB(x)-f} \mu_{f'->x}$
- Each factor-node marginalizes the messages it receives from all-but-one neighboring variable-node, and sends the product to remaining variable-node $\mu_{f->x} = \prod_{x':NB(f)-x} \sum_{x'} \mu_{x'->f}$

Approximate Inference

Approximate Inference

- In many cases, exact inference by above processes not feasible
- There may be too many variables, coupled in a complex way
- There may be continuous variables with intractable integration
- Aim: calculate $p(Z|X)$ where Z are latent, X are observed
- Two main approaches:
  - Draw samples (directly or indirectly) to construct the distribution
  - Create another tractable distribution for $Z|X$ and choose its parameters so that it resembles the original one

Sampling-based Inference

- Assumption: joint probability $p(X, Z)$ can be calculated point-wise
- Aim: calculate posterior distribution $p(Z|X)$, which cannot be done directly (cannot calculate $p(X)$)
- Approach: draw samples of Z from $p(Z|X)$, store them and construct the posterior from them
- But drawing samples itself maybe difficult (especially if Z includes many variables)
- Approach: *Markov Chain Monte Carlo* sampling!
- Instead of of drawing samples from $p(Z|X)$, explore the sample space using Markov Chain transitions
- Stationary distribution of the Markov Chain should be the desired distribution!
- Needed: given a particular sample, a *transition distribution* to jump to another sample!

Gibbs Sampling

- A special category of Markov Chain Monte Carlo
- Sample transition restricted: only one variable of Z may be changed at each transition
- Explore sample space by moving along one dimension at a time
- While updating variable Z_k, the other variables Z_{-k} are held unchanged
- We need distribution $p(Z_k|Z_{-k}, X)$, which should be easy to calculate and sample from
  - Initialize Z (using any reasonable approach)
  - Repeat for each variable $Z_k \in Z$:
  - Sample $Z_k \sim p(Z_k|Z_{-k}, X)$, update Z_k with newly sampled value
  - Repeat the above process many items, keep storing Z at regular intervals (usually after 5 or 10 iterations)
  - When sufficient number of samples stored, estimate posterior on Z based on those

Example of Gibbs Sampling

- Consider a Hidden Markov Model with known parameters
- Aim: calculate posterior distribution of state sequence Z
- Required: $p(Z(t)|Z(-t), X)$ where $Z(-t) = \{Z(1), \ldots, Z(t-1), Z(t+1), \ldots, Z(T)\}$
- $p(Z(t)|Z(-t), X) \propto p(Z(t)|Z(t-1)) * p(Z(t+1)|Z(t)) * p(X(t)|Z(t))$ (other terms independent of $Z(t)$)
- $X(t), Z(t-1), Z(t+1)$ are observed or fixed
- Easy to sample $Z(t)$ now!
- If parameters are unknown, they can be initialized and updated after each round of Gibbs Sampling (based on temporary Z-values)
- *Collapsed Gibbs Sampling*: when some variables are marginalized out of original distribution
- Advantage: easy to implement, can give accurate results
- Disadvantage: too slow, needs thousands of samples

Approximating a distribution

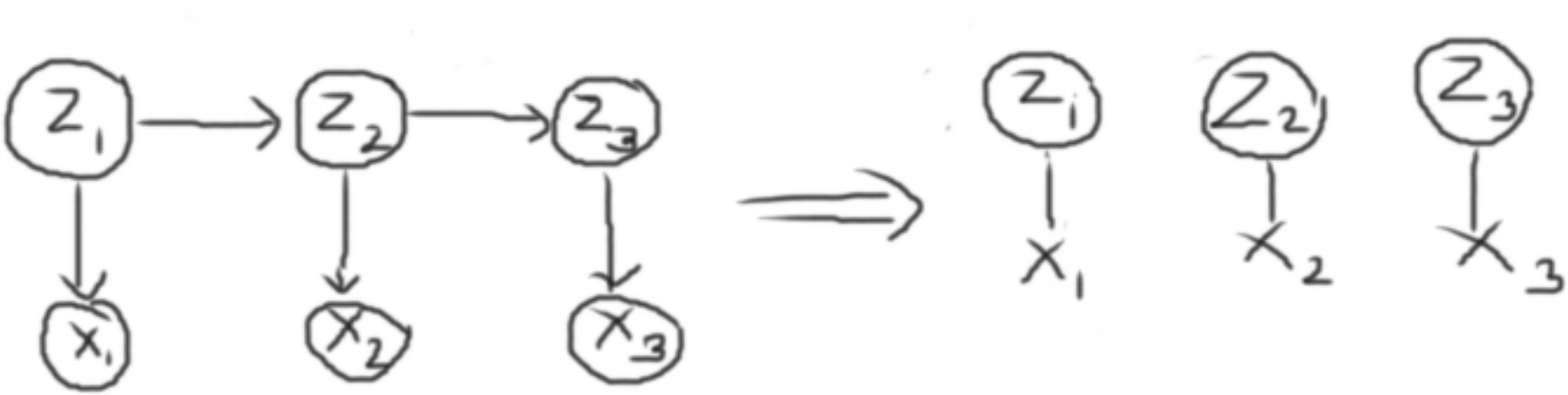
- Given joint distribution $p(Z, X)$, we need $p(Z|X)$
- But we do not know $p(X)$: marginalization intractable
- Define a proposal distribution $q(Z)$ that is easy to compute
- Aim: minimize KL-divergence between $q(Z)$ and $p(Z, X)$
- Kullback-Leibler (KL) Divergence between distributions p and q: $KL(p||q) = \sum_x p(x) log(\frac{p(x)}{q(x)})$
- Can also be considered as expectation of $log(p/q)$ with respect to p
- Always non-negative, equal to 0 iff $p = q$, high values for dissimilar distributions
- Also, $KL(p||q) \neq KL(q||p)$ (assymmetric relation)

Evidence Lower Bound (ELBO)

- Define $J(q) = KL(q(Z)||p(X,Z)) = \sum_Z q(Z) log(\frac{q(Z)}{p(X,Z)})$
- i.e. $J(q) = \sum_Z q(Z)(log(\frac{q(Z)}{p(Z|X)}) - log(p(X)) = \sum_Z q(Z)(log(\frac{q(Z)}{p(Z|X)})) - log(p(X))$
- i.e. $J(q) = KL(q(Z)||p(Z|X)) - log(p(X))$
- As K-L divergence is non-negative, it follows that $logp(X) \geq -J(q)$
- In other words, $-J(q)$ is the lower-bound for the evidence $p(X)$ (Evidence Lower Bound- ELBO)
- Minimize $J(q)$ is equivalent to maximizing the evidence lower bound, a proxy for the evidence itself
- It can also be seen as minimizing $KL(q(Z)||p(Z|X))$, i.e. pushing q towards the desired posterior distribution $p(Z|X)$

Variational Inference and Mean-Field Approximation

- General approach: 1. Choose q, 2. Define ELBO 3. Optimize parameters of q to maximize ELBO
- Popular choice for q: fully factored distribution $q(Z) = \prod_i q_i(Z_i)$
- For each factor i, optimize q_i holding other factors q_{-i} unchanged
- Solution: $log(q_i(Z_i)) = E_{-q_i} log(p(X, Z)) + constant$, i.e. calculate expectation of the joint distribution using same value of Z_i, over the remaining variables of q except Z_i
- Normalize to eliminate constant terms, iterate over all variables till convergence



Example of Mean-Field Approximation

- Hidden Markov Model: $p(Z, X) = p(Z_1) * \prod_{t=2}^{T} p(Z_t|Z_{t-1}) * \prod_{t=1}^{T} p(X_t|Z_t)$
- Target $p(Z|X)$, variational approximation $q(Z) = \prod_{t=1}^{T} q_t(Z_t|\pi_t)$
- Z_t at every t follows independent Categorical distribution with parameter π_t!
- Variation update: $log(q_t(Z_t)) = E_{-q_t} log(p(Z, X)) + c_t$
- Consider only those terms of $p(Z, X)$ that involve Z_t, i.e. $log(p(Z_{t+1}|Z_t)) + log(p(Z_t|Z_{t-1})) + log(p(X_t|Z_t))$
- Take expectation of $log(p(Z_{t+1}|Z_t))$ over $q_{t+1}(Z_{t+1})$, and $log(p(Z_t|Z_{t-1}))$ over $q_{t-1}(Z_{t-1})$
- $log(p(Z_{t+1}|Z_t)) = \sum_{i,j} I(Z_t = i)I(Z_{t+1} = j)log(A_{ij})$
- Taking expectation w.r.t. $q_{t+1}(Z_{t+1})$ yields $\pi_{t+1,j}$, i.e. the probability that $Z_{t+1} = j$ holds under $q_{t+1}(Z_{t+1})$
- Similarly calculate expectation of $log(p(Z_t|Z_{t-1}))$ w.r.t. $q_{t-1}(Z_{t-1})$
- Multiply these factors and normalize to get $q_t(Z_t)$

Thank you!