10-708 PGM (Spring 2019): Homework 1 v1.1

Andrew ID: [your Andrew ID]
Name: [your first and last name]
Collaborators: [Andrew IDs of all collaborators, if any]

1 Bayesian Networks [20 points] (Xun)

State True or False, and briefly justify your answer in a few sentences. You can cite theorems from Koller and Friedman (2009). Throughout the section, P is a distribution and $\mathcal{G}$ is a BN structure.

1. [**2 points**] If $A \perp B \mid C$ and $A \perp C \mid B$, then $A \perp B$ and $A \perp C$. (Suppose the joint distribution of A, B, C is positive.)

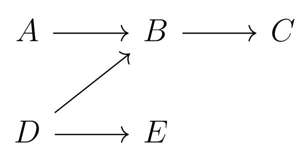


Figure 1: A Bayesian network.

2. [**2 points**] In Figure 1, $E \perp C \mid B$.
3. [**2 points**] In Figure 1, $A \perp E \mid C$.

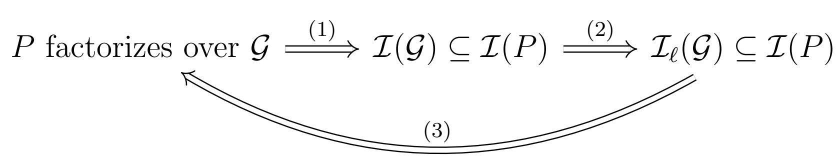


Figure 2: Some relations in Bayesian networks.

4. [**2 points**] In Figure 2, relation (1) is true.
5. [**2 points**] In Figure 2, relation (2) is true.
6. [**2 points**] In Figure 2, relation (3) is true.

7. [**2 points**] If $\mathcal{G}$ is an I-map for P, then P may have extra conditional independencies than $\mathcal{G}$.
8. [**2 points**] Two BN structures $\mathcal{G}_1$ and $\mathcal{G}_2$ are I-equivalent iff they have the same skeleton and the same set of v-structures.
9. [**2 points**] The minimal I-map of a distribution is the I-map with fewest edges.
10. [**2 points**] The P-map of a distribution, if exists, is unique.

2 Undirected Graphical Models [25 points] (Paul)

2.1 Local, Pairwise and Global Markov Properties [18 points]

1. Prove the following properties:
   - [**2 points**] If $A \perp (B, D) \mid C$ then $A \perp B \mid C$.
   - [**2 points**] If $A \perp (B, D) \mid C$ then $A \perp B \mid (C, D)$ and $A \perp D \mid (B, C)$.
   - [**2 points**] For strictly positive distributions, if $A \perp B \mid (C, D)$ and $A \perp C \mid (B, D)$ then $A \perp (B, C) \mid D$.
2. [**6 points**] Show that for any undirected graph G and distribution P, if P factorizes according to G, then P will also satisfy the global Markov properties of G.
3. [**6 points**] Show that for any undirected graph G and distribution P, if P satisfies the local Markov property with respect to G, then P will also satisfy the pairwise Markov property of G.

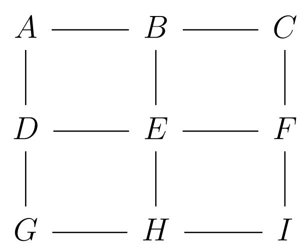
2.2 Gaussian Graphical Models [7 points]

Now we consider a specific instance of undirected graphical models. Let $X = \{X_1, ..., X_d\}$ be a set of random variables and follow a joint Gaussian distribution $X \sim \mathcal{N}(\mu, \Lambda^{-1})$ where $\Lambda \in \mathbb{S}^{++}$ is the precision matrix. Let X_j, X_k be two nodes in X, and $Z = \{X_i \mid i \notin \{j, k\}\}$ denote the remaining nodes. Show that $X_j \perp X_k \mid Z$ if and only if $\Lambda_{jk} = 0$.

3 Exact Inference [40 points] (Xun)

3.1 Variable elimination on a grid [10 points]

Consider the following Markov network:

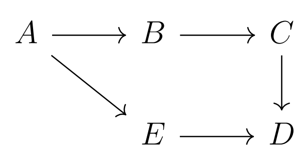


We are going to see how *tree-width*, a property of the graph, is related to the intrinsic complexity of variable elimination of a distribution.

1. [**2 points**] Write down largest clique(s) for the elimination order $E, D, H, F, B, A, G, I, C$.
2. [**2 points**] Write down largest clique(s) for the elimination order $A, G, I, C, D, H, F, B, E$.
3. [**2 points**] Which of the above ordering is preferable? Explain briefly.
4. [**4 points**] Using this intuition, give a reasonable ($\ll n^2$) upper bound on the tree-width of the $n \times n$ grid.

3.2 Junction tree in action: part 1 [10 points]

Consider the following Bayesian network $\mathcal{G}$:



We are going to construct a junction tree $\mathcal{T}$ from $\mathcal{G}$. Please sketch the generated objects in each step.

1. [**1 pts**] Moralize $\mathcal{G}$ to construct an undirected graph $\mathcal{H}$.
2. [**3 pts**] Triangulate $\mathcal{H}$ to construct a chordal graph $\mathcal{H}^*$.

   (Although there are many ways to triangulate a graph, for the ease of grading, please use the triangulation that corresponds to the elimination order A, B, C, D, E.)

3. [**3 pts**] Construct a cluster graph $\mathcal{U}$ where each node is a maximal clique $\boldsymbol{C}_i$ from $\mathcal{H}^*$ and each edge is the sepset $\boldsymbol{S}_{i,j} = \boldsymbol{C}_i \cap \boldsymbol{C}_j$ between adjacent cliques $\boldsymbol{C}_i$ and $\boldsymbol{C}_j$.

4. [**3 pts**] Run maximum spanning tree algorithm on $\mathcal{U}$ to construct a junction tree $\mathcal{T}$.

   (The cluster graph is small enough to calculate maximum spanning tree in one's head.)

3.3 Junction tree in action: part 2 [20 points]

Continuing from part 1, now assume all variables are binary and the CPDs are parameterized as follows:

A	$P(A)$
0	x_0

A	B	$P(B\|A)$
0	0	x_1
1	0	x_2

A	E	$P(E\|A)$
0	0	x_3
1	0	x_4

B	C	$P(C\|B)$
0	0	x_5
1	0	x_6

C	E	D	$P(D\|C,E)$
0	0	0	x_7
0	1	0	x_8
1	0	0	x_9
1	1	0	x_{10}

We are going to implement belief propagation on $\mathcal{T}$. The provided template `junction_tree.py` contains the following tasks:

- `initial_clique_potentials()`: Compute initial clique potentials $\psi_i(\boldsymbol{C}_i)$ from factors ϕ_i.
- `messages()`: Compute messages $\delta_{i\to j}$ from initial clique potentials $\psi_i(\boldsymbol{C}_i)$.
- `beliefs()`: Compute calibrated clique beliefs $\beta_i(\boldsymbol{C}_i)$ and sepset beliefs $\mu_{i,j}(\boldsymbol{S}_{i,j})$, using initial clique potentials $\psi_i(\boldsymbol{C}_i)$ and messages $\delta_{i\to j}$.
- Using the beliefs $\beta_i(\boldsymbol{C}_i), \mu_{i,j}(\boldsymbol{S}_{i,j})$, compute
  - `query1()`: $P(B)$
  - `query2()`: $P(A|C)$
  - `query3()`: $P(A, B, C, D, E)$

Please finish the unimplemented TODO blocks and submit completed `junction_tree.py` to Gradescope (`https://www.gradescope.com/courses/36025`).

In the implementation, please represent factors as `numpy.ndarray` and store different factors in a dictionary with its scope as the key. For example, as provided in the template, `phi['ab']` is a factor ϕ_{AB} represented as a 2×2 matrix, where `phi['ab'][0, 0]` $= \phi_{AB}(A = 0, B = 0) = P(B = 0|A = 0) = x_1$. For messages, one can use `delta['ab_cd']` to denote a message from AB to CD. Most functions can be written in 3 lines of code. You may find `np.einsum()` useful.

4 Parameter Learning [15 points] (Xun)

$$
\begin{array}{ccccccc}
Y_1 & \longrightarrow & Y_2 & \longrightarrow & \cdots & \longrightarrow & Y_T \\
\downarrow & & \downarrow & & & & \downarrow \\
X_1 & & X_2 & & & & X_T
\end{array}
$$

Consider an HMM with $Y_t \in [M]$, $X_t \in \mathbb{R}^K$ $(M, K \in \mathbb{N})$. Let $(\pi, A, \{\mu_i, \sigma_i^2\}_{i=1}^M)$ be its parameters, where $\pi \in \mathbb{R}^M$ is the initial state distribution, $A \in \mathbb{R}^{M \times M}$ is the transition matrix, $\mu_i \in \mathbb{R}^K$ and $\sigma_i^2 > 0$ are parameters of the emission distribution, which is defined to be an isotropic Gaussian. In other words,

$$P(Y_1 = i) = \pi_i \tag{1}$$

$$P(Y_{t+1} = j | Y_t = i) = A_{ij} \tag{2}$$

$$P(X_t | Y_t = i) = \mathcal{N}(X_t; \mu_i, \sigma_i^2 I). \tag{3}$$

We are going to implement the Baum-Welch (EM) algorithm that estimates parameters from data $\boldsymbol{X} \in \mathbb{R}^{N \times T \times K}$, which is a collection of N observed sequences of length T. Note that there are different forms of forward-backward algorithms, for instance the (α, γ)-recursion, which is slightly different from the (α, β)-recursion we saw in the class. For the ease of grading, however, please implement the (α, β) version, and remember to normalize the messages at each step for numerical stability.

Please complete the unimplemented TODO blocks in the template `baum_welch.py` and submit it to Gradescope (`https://www.gradescope.com/courses/36025`). The template has its own toy problem to verify the implementation. The test cases are ran on other randomly generated problem instances.

References

D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.