We present algorithms for Bayesian learning of decomposable models from data. Priors of a certain form admit exact averaging in $O(3^n n^3)$ time and sampling $T$ graphs from the posterior in $O(4^n + nT)$ time. To target a broader class of priors we associate each sample with an importance weight. Empirically, we compare averaging to optimization and demonstrate the accuracy of our importance sampling estimates.

**PROBLEM**

**DECOMPOSABLE MODEL**

- Undirected graph $G$ on $V = \{1, \ldots, n\}$ with a junction tree.
- A joint distribution $p(x_1, \ldots, x_n)$ that factorizes as
  $$p(x_V) = \prod_{C \in \mathcal{C}} p(x_C) \prod_{S \in \mathcal{S}} p(x_S),$$
  where $\mathcal{C}$ is the set of cliques of $G$, $\mathcal{S}$ is the multiset of separators, $x_C = (x_v : v \in C)$.

**EXACT AVERAGING**

Computing $Z_\psi$ for a decomposable $\psi$ is doable up to $n \approx 8$. It is more efficient to compute
$$Z_{\psi_T} = \sum_G \tau(G) \psi(G),$$
where $\tau(G)$ denotes the number of rooted junction trees of $G$. We get $Z_{\psi_T} = f(V, G)$, where $f$ is obtained by adapting a recent dynamic programming algorithm [1]:

1. Given sets $S$ and $R$, choose a clique $C$ within $R$ that contains $S$ completely:
   $$f(S, R) = \sum_{C \subseteq S \cup R} \psi(C) g(C, R \setminus C).$$

2. Partition the remaining vertices into $R_1, \ldots, R_k$:
   $$g(C, U) = \sum_{\min U \subseteq R} h(C, R) g(C, U \setminus R).$$

3. For each part $R$ choose a separator $S$ contained in $C$. Recurse back to step 1:
   $$h(C, R) = \frac{f(S, R)}{\psi(S)}.$$

The algorithm runs in $O(4^n)$ time and $O(3^n)$ space. By using variants of fast zeta transform and fast subset convolution, we can bring the running time to $O(3^n n^3)$.

**MONTE CARLO AVERAGING**

To estimate the posterior expectation $Z_{\psi_T}$:
1. Run the exact algorithm for a $\psi \propto \pi$.
2. Backtrack into the recurrences to draw samples $G_1, \ldots, G_T$ from $\pi \propto \psi_T$.
   - (a) Fixed bucketing: tunable time/space tradeoff between $O(2^n)$ and $O(n)/O(4^n)$; or
   - (b) Adaptive caching: good if $\pi$ is skewed.
3. Obtain the self-normalized importance sampling estimate
   $$\tilde{Z}_{\psi_T} = \frac{1}{\tau(G_i)} \sum_{t=1}^T w_t \psi(G_t), w_t = \frac{1}{\tau(G_t)}.$$

Computing $\tau(G_t)$ takes $O(n^2)$ time [2].

**EXPERIMENTS**

We ran our implementation [3] on datasets of a varying number of variables ($n$) and records ($m$). The table shows the seconds spent in the exact algorithm (E) and sampling $10^2$ graphs (S).

<table>
<thead>
<tr>
<th>Dataset</th>
<th>$n$</th>
<th>$m$</th>
<th>$E$</th>
<th>$S$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Asia</td>
<td>8</td>
<td>1000</td>
<td>0.018</td>
<td>33</td>
</tr>
<tr>
<td>Bridges</td>
<td>12</td>
<td>108</td>
<td>5.6</td>
<td>54</td>
</tr>
<tr>
<td>Flare</td>
<td>13</td>
<td>1066</td>
<td>24</td>
<td>77</td>
</tr>
<tr>
<td>Votes</td>
<td>17</td>
<td>435</td>
<td>7497</td>
<td>167</td>
</tr>
</tbody>
</table>

**BAYESIAN LEARNING**

Problem: Compute the marginal $Z_\psi = \sum_G \psi(G)$, where $\psi$ is a decomposable function, i.e.,
$$\psi(G) = \prod_{C \in \mathcal{C}} \psi(C) \prod_{S \in \mathcal{S}} \psi(S), \quad \psi(C, \psi_S : 2^V \to \mathbb{R}.$$

Example (feature posterior probability): The posterior probability of a feature $\psi(G)$ is the expectation $E(\psi(G) \mid data) = Z_{\psi_T}$ where $\pi(G) \propto Pr(data \mid G) Pr(G)$.

Prediction accuracy – averaging vs. MAP: The log probability (y-axis) of a test set for a varying amount of training data (x-axis).

Edge posterior probabilities: The largest estimation error over all edges (y-axis) for a varying number of samples (x-axis).