Lab 7: Gibbs Sampling for Inference

INF581: Advanced Machine Learning and Autonomous Agents

Abstract

This week's lab involves two tasks; the first task, which involves Gibbs sampling for multi-output inference, is graded, and the second task, about imitation learning and inverse reinforcement learning, is a bonus task, designed as an introduction to these topics.

Completion instructions: For the first task, read the remainder of this document, and complete the python file `conditional_dependency_network.py` and submit it. For the second (bonus) task, please see the file `Lab_IRL.ipynb` for instructions. It can be submitted *to the same link* as the first task.

Suppose the model in Fig. 1.

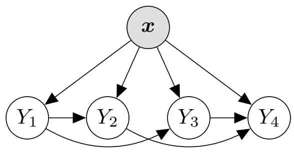

Figure 1: Bayesian Network.

Estimating $P(\boldsymbol{y} \mid \boldsymbol{x})$ is not a problem if we have morginals $P(Y_j|\boldsymbol{x}, \ldots)$. Although we cannot explore all combinations of $\boldsymbol{y} \in \{0,1\}^m$ (as we did in Week 1) when $m \gg 4$, we could use *Monte Carlo ancestral sampling* (as suggested in Week 2):

$$
\begin{aligned}
y_1^{(t)} &\sim P(Y_1|\boldsymbol{x}) \\
y_2^{(t)} &\sim P(Y_2|\boldsymbol{x}, y_1^{(t)}) \\
y_3^{(t)} &\sim P(Y_3|\boldsymbol{x}, y_1^{(t)}) \\
y_4^{(t)} &\sim P(Y_4|\boldsymbol{x}, y_1^{(t)}, y_3^{(t)})
\end{aligned}
$$

to collect samples $\{\boldsymbol{y}^{(1)}, \ldots, \boldsymbol{y}^{(T)}\}$. This even works when labels are in the continuous domain.

But now, suppose, Fig. 2, and specifically Fig. 2 (mid). In this case the observed nodes are descents of the unobserved nodes, so ancestral sampling will note work. This could occur, for example: in a recommendation system ($y_j = 1 \Leftrightarrow$ user $\boldsymbol{x}$ 'like's the j-th product) where some [but not all] products have already been 'like'd; or in missing value imputation (some input values have not been observed, and we wish to fill them in); or when an expert has partially intervened a decision making process; providing some, but not all, of the labels.

In Fig. 2 (right), we have converted the graph into a conditional dependency network (CDN) which indicates how Gibbs sampling can proceed, sampling in an *undirected* fashion.

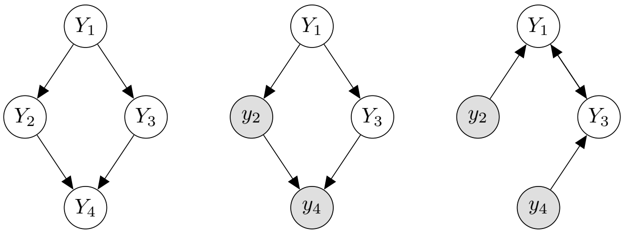

Figure 2: Left: As in Fig. 1, simply with $\boldsymbol{x}$ not shown (and Y-nodes visually rearranged) for clarity; Mid: where y_2 and y_4 are already observed as evidence. Right: converted into a CDN by considering the Markov blanket (the children, parents, and parents of children of Y_1 include: $\{y_2, Y_3\}$; etc.

The code should be completed in `conditional_dependency_network.py`, via:

- Implement the Markov blanket function (`_markov_blanket`); thus allowing the automatic conversion of the Bayesian network into a CDN
- Implement Gibbs sampling (the function `gibbs_sampling`)
- Report mode and marginal point estimates, alongside joint and marginal distribution estimates (the function `predict_proba_x`).

We make the coding/implementation a little more straightforward by assuming that all observed nodes can be considered as $\boldsymbol{x}$ (e.g., $\boldsymbol{x} \equiv [y_2, y_4]$ wrt Fig. 2 (right)).

As usual, do not change any function's or parameter's name, and do not import any extra module.

Upload the file `conditional_dependency_network.py` to the submission/file drop-off box indicated in Moodle.