Problem Set 3 Solutions

6.S091: Causality

IAP 2023

Problem 1: Constructing Minimal I-MAPs [5 points]

Question

You should use the partial-correlation based conditional independence test which you coded in Problem Set 2. An implementation of the necessary functions can be found at

`https://github.com/csquires/6.S091-causality/blob/main/psets/pset3/utils.py`

(a) Write a function

`minimal_map(samples, permutation, alpha)`

which constructs the graph $\mathcal{G}_\pi$ for the `permutation` π, using the partial-correlation based conditional independence test with significance level `alpha`.

Samples of $(X_1, X_2, X_3, X_4, X_5)$ are in the file `imap_samples.csv` at

`https://github.com/csquires/6.S091-causality/blob/main/psets/pset3/imap_samples.csv`

Run `minimal_imap(imap_samples, permutation, 0.05)` for the following permutations:

$$\pi_a = [1, 2, 3, 4, 5]$$
$$\pi_b = [5, 4, 1, 2, 3]$$
$$\pi_c = [5, 4, 3, 2, 1]$$

The graph $\mathcal{G}_{\pi_a}$ should have 4 edges. Draw the graphs for each permutation π_a, π_b, and π_c.

(b) Recall that a Chickering sequence from $\mathcal{G}$ to $\mathcal{H}$ is a sequence of DAGs, where each consecutive pair in the sequence is related by a covered edge reversal or an edge addition. Construct a Chickering sequence from $\mathcal{G}_{\pi_a}$ to $\mathcal{G}_{\pi_b}$. You can do this problem by hand, you do not need to write a function.

Answer

(a) See Figure 1.

(b) See Figure 2.

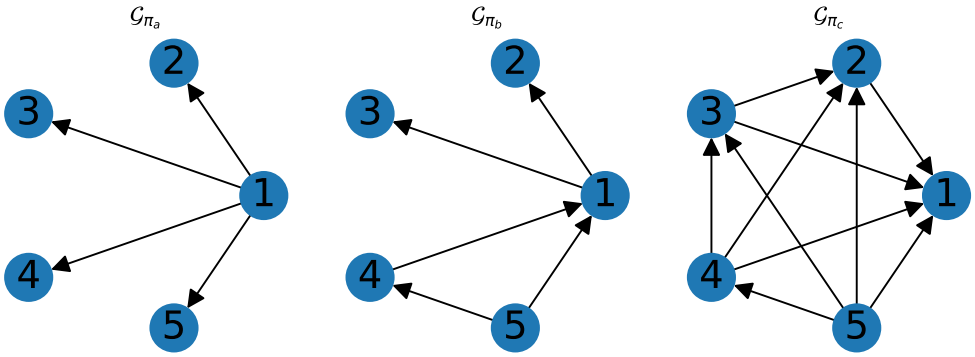

Figure 1: Minimal IMAPs

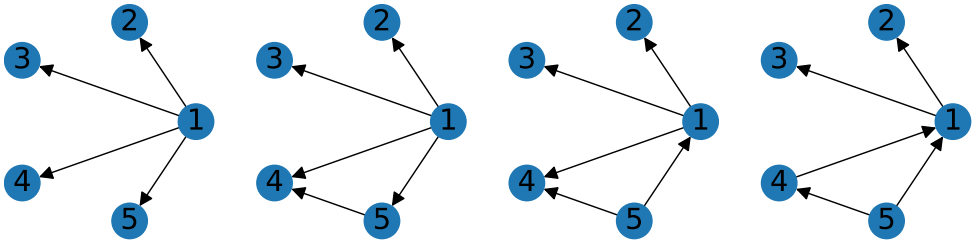

Figure 2: Chickering sequence

Problem 2: d-separation defines a graphoid [5 points]

Questions

In this problem, you will show that two of the graphoid properties hold for d-separation. Throughout the problem, let $\mathbf{A}, \mathbf{B}_1, \mathbf{B}_2, \mathbf{S} \subseteq \mathcal{X}$ be disjoint subsets of nodes in a DAG $\mathcal{G}$, and let $\mathbf{B} = \mathbf{B}_1 \cup \mathbf{B}_2$.

(a) Weak Union. Suppose there is a d-connecting path γ from $A \in \mathbf{A}$ to $B \in \mathbf{B}_1$ given $\mathbf{S} \cup \mathbf{B}_2$. Show that there is a d-connecting path γ' from some $A' \in \mathbf{A}$ to some $B' \in \mathbf{B}$ given $\mathbf{S}$.

(b) Contraction. Suppose there is a d-connecting path γ from $A \in \mathbf{A}$ to $B \in \mathbf{B}$ given $\mathbf{S}$. Show that either:

- There is a d-connecting path γ' from some $A' \in \mathbf{A}$ to some $B' \in \mathbf{B}_2$ given $\mathbf{S}$, or
- There is a d-connecting path γ' from some $A' \in \mathbf{A}$ to some $B' \in \mathbf{B}_1$ given $\mathbf{S} \cup \mathbf{B}_2$.

Answers

(a) Weak Union. Let γ_k be a non-collider in γ. Then $\gamma_k \notin \mathbf{S}$, so γ_k is unblocked given $\mathbf{S}$. Let γ_k be a collider in γ. Then either

(i) $\overline{\text{de}}_{\mathcal{G}}(\gamma_k) \cap \mathbf{S} \neq \varnothing$, or

(ii) $\overline{\text{de}}_{\mathcal{G}}(\gamma_k) \cap \mathbf{S} = \varnothing$, but $\overline{\text{de}}_{\mathcal{G}}(\gamma_k) \cap \mathbf{B}_2 \neq \varnothing$.

Let γ_k be the collider on γ closest to A for which case (ii) holds, if any such collider exists. Let γ' be the concatenation of γ_1, the segment of γ from A to γ_k, and γ_2, a shortest path from γ_k to some descendant B' in $\mathbf{B}_2$. Then γ' is d-connecting between A and B': it is unblocked at all nodes in γ_1, and at all nodes in γ_2, since all nodes in γ_2 are non-colliders which are not in $\mathbf{S}$.

(b) Contraction. If $B \in \mathbf{B}_2$, then γ_k is a d-connecting path from $A \in \mathbf{A}$ to $B' = B \in \mathbf{B}_2$ given $\mathbf{S}$. Otherwise, if γ_k is a collider in γ, then $\overline{\text{de}}_{\mathcal{G}}(\gamma_k) \cap \mathbf{S} \neq \varnothing$, so γ_k is unblocked given $\mathbf{S} \cup \mathbf{B}_2$. If γ_k is a non-collider in γ, it is not in $\mathbf{S}$, so either (i) it is unblocked given $\mathbf{S} \cup \mathbf{B}_2$, or (ii) we have $\gamma_k \in \mathbf{B}_2$. Let γ_k be the non-collider on γ closest to A for which case (ii) holds, if any such non-collider exists. If such a non-collider exists, then γ' equal to the segment of γ from A to γ_k is a d-connecting path from A to $B' = \gamma_k \in \mathbf{B}_2$. Otherwise, γ is a d-connecting path from A to B.

Problem 3: Searching an Equivalence Class [5 points]

Question

The folder

`https://github.com/csquires/6.S091-causality/tree/main/psets/pset3/mec_examples`

contains pairs of the form

`starting_dag1.csv` `ending_dag1.csv`

where the starting DAG $\mathcal{G}$ is Markov equivalent to the ending DAG $\mathcal{H}$. Each file represents a DAG in adjacency matrix form, i.e., each file stores a matrix A where $A_{ij} = 1$ if and only if the associated DAG $\mathcal{G}$ has an edge $i \to j$. Starter code for loading DAGs can be found at

`https://github.com/csquires/6.S091-causality/blob/main/psets/pset3/search_mec.py`

(a) Write a function

`covered_edge_neighbors(dag)`

which, given a DAG $\mathcal{G}$, returns all DAGs $\mathcal{G}'$ which differ from $\mathcal{G}$ by exactly one covered edge reversal. `starting_dag1` should have 5 neighbors. How many neighbors do `starting_dag2` and `starting_dag3` have?

(b) Write a function

`search_mec(starting_dag, ending_dag)`

which takes two Markov equivalent DAGs $\mathcal{G}$ and $\mathcal{H}$ and returns a sequence of covered edge reversals from $\mathcal{G}$ to $\mathcal{H}$.

The sequence of covered edges can be found by breadth-first search using `covered_edge_neighbors`. For breadth-first search, you might need to maintain a set of already-visited states. In Python, I recommend using a `frozenset` containing the edges of a DAG to represent each state, which is hashable and hence can be used in a set or as keys of a dictionary.

The shortest path from `starting_dag1` to `ending_dag1` should be of length 6 (where the length of the path is the number of covered edge flips). How long is the shortest path from `starting_dag2` to `ending_dag2`? How long is the shortest path from `starting_dag3` to `ending_dag3`?

Answer

(a) `starting_dag2` has 4 covered edges, and thus 4 neighbors. `starting_dag3` also has 4 covered edges, and thus 4 neighbors.

(b) The shortest path from `starting_dag2` to `ending_dag2` is 1 covered edge flip. The shortest path from `starting_dag3` to `ending_dag3` is 3 covered edge flips.