6.S091: Problem Set 3

Hyewon Jeong
hyewonj@mit.edu

1 Constructing Minimal I-MAPs [5 points]

You should use the partial-correlation based conditional independence test which you coded in Problem Set 2.

(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 texttttimap_samples.csv. Run `minimal_map(samples, permutation, alpha)` for the following permutations:

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

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

Answer
Graphs for each permutation is drawn as below in Figure 1: π_a has 4 edges, π_b has 10 edges, and π_c has 5 edges. The code used for generating graph (`minimal_map`) is in Figure 1d.

(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 $\mathcal{G}_{\pi_c}$. You can do this problem by hand, you do not need to write a function.
A Chickering sequence from $\mathcal{G}_{\pi_a}$ to $\mathcal{G}_{\pi_c}$ is shown in Figure 2. We can first start by adding an edge $(5 \to 4)$to $\mathcal{G}_{\pi_a}12345$ to make $\mathcal{G}12354$, then we reverse the covered edge $1 \to 5$ to make $\mathcal{G}51423$. Finally we reverse another covered edge $1 \to 4$ to make $\mathcal{G}54123 = \mathcal{G}_{\pi_c}$.

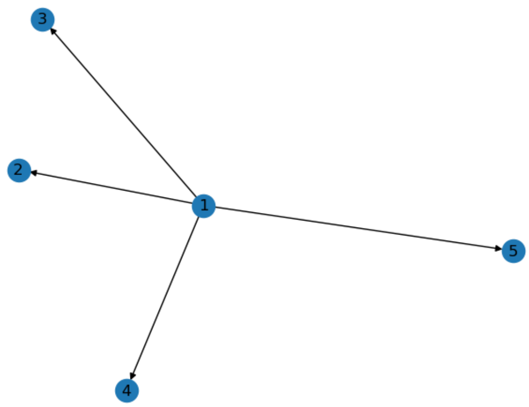
2 d-separation defines a graphoid [5 points]

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 $\mathbf{B} = \mathbf{B}_1 \cup \mathbf{B}_2$.

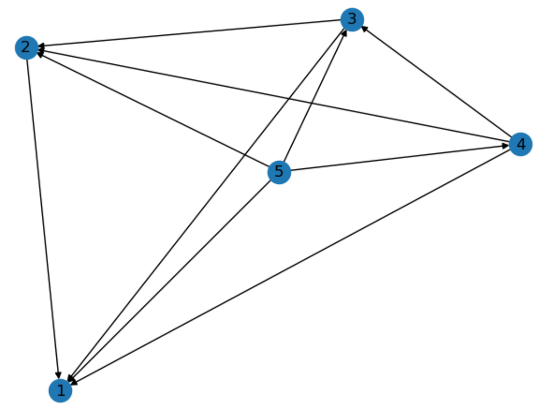
(a) Weak Union. Suppose there is a d-connecting path γ from $A \in \mathbf{A}$ to $B \in \mathbf{B}_1$ given $\mathbf{S} \cup \mathbf{B}_2$. Show that there is a d-connecting path γ' from some $A' \in \mathbf{A}$ to some $B' \in \mathbf{B}$ given $\mathbf{S}$.

Answer
Let γ_k be an arbitrary node in a d-connecting path γ between $A \in \mathbf{A}$ and $B \in \mathbf{B}_1$ given $\mathbf{S} \cup \mathbf{B}_2$ ($\mathbf{A} \not\perp\!\!\!\perp \mathbf{B}_1 | \mathbf{S} \cup \mathbf{B}_2$). If γ_k is a collider in a path, then in order for it to be unblocked, $\gamma_k \in S \cup B_2$. If $\gamma_k \in S$, then it follows that the path is a d-connecting path from $A \in \mathbf{A}$ to $B \in \mathbf{B}_1 \subset \mathbf{B}$ that $\mathbf{A} \not\perp\!\!\!\perp \mathbf{B} | \mathbf{S}$. Next, if $\gamma_k \in B_2$ then the path from A to γ_k becomes the d-connecting path γ' from $A \in \mathbf{A}$ to $\gamma_k \in \mathbf{B}_2 \subset \mathbf{B}$. Thus, in the case where γ_k is a collider, it follows that there is a d-connecting path γ' such that $\mathbf{A} \not\perp\!\!\!\perp \mathbf{B} | \mathbf{S}$.

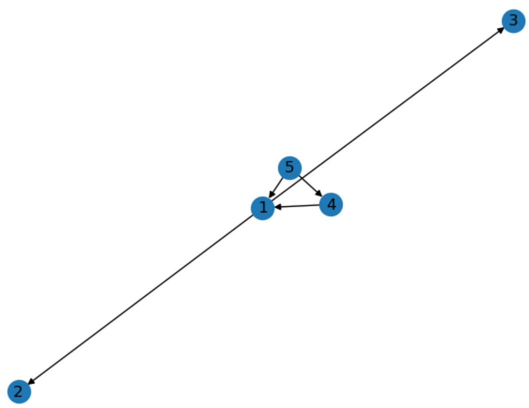
Next, if γ_k is a non-collider, then $\gamma_k \notin S \cup B_2$ and this means that γ_k is unblocked and thus path from $A \in \mathbf{A}$ and $B \in \mathbf{B}_1 \subset \mathbf{B}$ is unblocked given S ($A \not\perp\!\!\!\perp B | S$). This is because $\gamma_k \notin S \cup B_2$ means $\gamma_k \notin S$ and that $B \in \mathbf{B}_1 \subset \mathbf{B}$. $\square$



(a) $\mathcal{G}_{\pi_a}$ 12345



(b) $\mathcal{G}_{\pi_b}$ 54321



(c) G_{π_c} 54123

```python
def minimal_map(samples, permutation, alpha):
    n = len(permutation)
    combination = list(itertools.combinations(permutation, 2))

    G = nx.DiGraph()
    G.add_nodes_from(permutation)

    for n1, n2 in combination:
        if permutation.index(n1) < permutation.index(n2):
            i = n1
            j = n2
        else:
            i = n2
            j = n1

        pre_j = [e for e in permutation if permutation.index(e) < permutation.index(j)]
        pre_j.remove(i)

        S = pre_j
        pvalue = compute_pvalue(samples, i, j, S)
        if pvalue < alpha:
            G.add_edge(i, j)
    return G
```

(d) code segment for `minimal_map`

Figure 1: Graph structure for each permutation with minimal map (I-MAP) method (d).

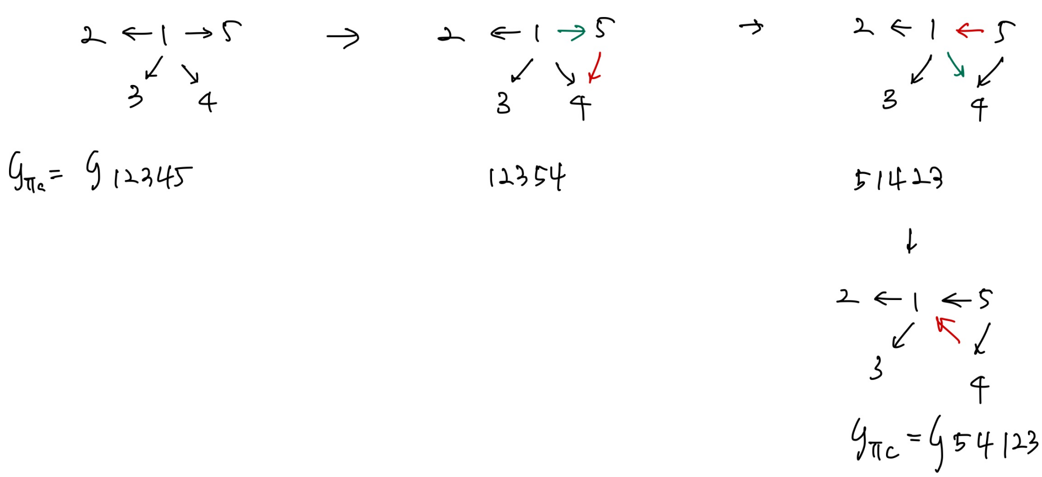


Figure 2: Chickering sequence from $\mathcal{G}_{\pi_a}$ to $\mathcal{G}_{\pi_c}$

(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 $\mathbf{S} \cup \mathbf{B}_2$.

Answer
We can consider two cases where $B \in \mathbf{B}_2$ and $B \in \mathbf{B}_1$:
First, if $B \in \mathbf{B}_2$, then there is a d-connecting path γ' from $A \in \mathbf{A}$ to $B \in \mathbf{B}_2$ given $\mathbf{S}$, meeting the condition of the first conclusion above.

If $B \in \mathbf{B}_1$ then let γ_k be an arbitrary node in a d-connecting path γ that spans from $A \in \mathbf{A}$ to $B \in \mathbf{B}$. We can consider two situations where γ_k is either collider or non-collider. If γ_k is a collider, then $\gamma_k \in \mathbf{S}$, which means $\gamma_k \in \mathbf{S} \cup \mathbf{B}_2$ and thus we can say the second case is true where there exists a d-connecting path γ' such that $\mathbf{A} \not\perp\!\!\!\perp \mathbf{B}_1 | \mathbf{S} \cup \mathbf{B}_2$. Next if γ_k is a non-collider, $\gamma_k \notin \mathbf{S}$ and we can think about two cases where $\gamma_k \in \mathbf{B}_2$ and $\gamma_k \notin \mathbf{B}_2$. In the first case, the path from A to $\gamma_k \in \mathbf{B}_2$ satisfies the first case of d-connecting path γ' that $A \in \mathbf{A}$ to $\gamma'_k \in \mathbf{B}_2$ given $\mathbf{S}$. In the second case, γ_k is not blocked thus we can say there exists a d-connecting path γ' with $\mathbf{A} \not\perp\!\!\!\perp \mathbf{B}_1 | \mathbf{S} \cup \mathbf{B}_2$. $\square$

3 Searching an Equivalence Class [5 points]

(a) `starting_dag2` and `starting_dag3` both have 4 neighbors. The code segment used for (a) and (b) is in Figure 3.

(b) The shortest path from `starting_dag2` to `ending_dag2`: 1
The shortest path from `starting_dag3` to `ending_dag3`: 3

```python
import numpy as np
import networkx as nx


def covered_edge_neighbors(G):
    covered_edges = []

    for i, j in G.edges:
        pa_i = [e for e in G.predecessors(i)]
        pa_j = [e for e in G.predecessors(j)]
        pa_i.append(i)

        if set(pa_i) == set(pa_j):
            covered_edges.append((i, j))

    return covered_edges

def search_mec(G, H):

    visited_dags = []
    queue = [(G, [])]

    while queue:
        curr_dag, curr_sequence = queue.pop(0)
        if curr_dag in visited_dags:
            continue

        visited_dags.append(curr_dag)

        if nx.is_isomorphic(curr_dag, H):
            return curr_sequence

        for i, j in covered_edge_neighbors(curr_dag):
            new_dag = curr_dag.copy()
            new_dag.remove_edge(i, j)
            new_dag.add_edge(j, i)
            queue.append((new_dag, curr_sequence + [(i, j)]))

    return None
```

Figure 3: Code segment for finding equivalence class