246 Course Project

Aryan Raj Dhawan

November 2024

1 Introduction

This paper will involve an analysis of different structural learning algorithms for Bayesian Networks. Given a dataset with attributes $V = v_1, ..., v_n$, we aim to learn a structure for a Bayesian network that best represents the relationships between these attributes. The metrics with which such a "best fit" structure is determined varies according to the type of algorithm employed for structure learning. In general, there are 3 primary types of structure learning algorithms:

1. Constraint-Based
2. Score-Based
3. Hybrid

The aim of this paper is two-fold. First, the theoretical and mathematical foundations of a few key algorithms from each category will be explored - highlighting both the differences between each category and between algorithms within each category as well. Second, the algorithms that are explored in the first section will be applied to a real-world problem of building the basis for a music recommender system.

2 Constraint-Based Algorithms

2.1 A broad overview

The broad idea of constraint-based algorithms is to find a **minimal I-map** for a distribution that satisfies a set of dependency (or independency) relationships. Prior to the discussion of the PC algorithm, a few key definitions must be introduced.

Definition 1. Given a distribution of random variables, an **I-Map** implies a subset of the conditional independence satisfied by the distribution (CITE MIT). Recall that a **P-Map** represents the independency relationships exactly. A **minimal I-Map** is one such that the removal of any edge would cease its property of being an I-Map.

Definition 2. The **Markov Condition** holds for a directed-acylic graph $G = (V, E)$ and a probability distribution over the vertices $P(V)$, if every variable $v \in V$ is conditionally independent of its (graphical) non-descendents, given its graphical parents. Formally,

$$\forall X, Y \in V, \forall Z \subseteq V \setminus \{X, Y\} : (X \perp\!\!\!\perp_d Y \mid Z \implies X \perp\!\!\!\perp_p Y \mid Z).$$

Definition 3. The **faithfulness** assumption extends this idea, and states that there are no additional independencies in the probability distribution $P(V)$ other than those dictated by the graph G. Formally, we get the condition below.

$$\forall X, Y \in V, \forall Z \subseteq V \setminus \{X, Y\} : (X \not\perp\!\!\!\perp_d Y \mid Z \implies X \not\perp\!\!\!\perp_p Y \mid Z).$$

Hence, if G implies no separation (graphically) between variables X and Y, there is no statistical independence between X and Y in $P(V)$. Together, the faithfulness and Markov conditions holding implies that

$$\forall X, Y \in V, \forall Z \subseteq V \setminus \{X, Y\} : (X \perp\!\!\!\perp_d Y \mid Z \iff X \perp\!\!\!\perp_p Y \mid Z).$$

This signifies that the dependency relationships in the data precisely match those in the graph G. Thus, faithfulness is assumed to "prove" that the learned graph is correct (CITE PPT)

2.2 The Peter Clark Algorithm

The Peter-Clark algorithm consists of 2 main parts - (1) skeleton recovery and (2) v-structure recovery.

The PC algorithm starts by assuming a fully connected graph H. During the **skeleton recovery** phase, the algorithm iteratively tests conditional independence relationships between pairs of nodes given subsets of other nodes. When $u \perp\!\!\!\perp_p v \mid S$ is found for a pair (u, v) and subset S, the edge $u - v$ is removed, and S is stored as the separation set S_{uv}. This process continues until the graph's structure "stabilizes". Note that to determine whether there is a set separating u and v, we might search all 2^{n-2} subsets of $V/\{u, v\}$. Hence, the complexity for investigating each possible edge in the skeleton is $O(2^n)$ and hence the complexity for constructing the skeleton is $O(n^2 * 2^n)$, where $n = |V|$.

In the **v-structure recovery** phase, the algorithm identifies and orients v-structures using the separation sets. A v-structure is oriented if u and v are not conditionally independent given w but share a common neighbor w. More formally, for all separation sets S_{uv} we consider $w \in V/(\{A, D\} \cup S_{AD})$ and if such a w exists, orient $u - w - v$ as $u \to w \leftarrow v$. Note that in many cases such a w may not exist, and thus the PC algorithm may return a **PDAG** - a partially directed acyclic graph - which contains some directed edges with the rest undirected (as they couldn't be oriented).

Thus, the overall time complexity of the PC algorithm is $O(n^{k+2})$, where $n = |V|$ and $k = \Delta G$ (the maximal degree of any vertex). This is owing to the number of conditional independence tests required by the algorithm (CITE).

Algorithm 1 The PC Algorithm for Learning DAGs

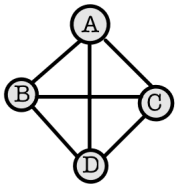
Input: A set V of nodes and a probability distribution P faithful to an unknown DAG G
Output: DAG pattern H
Initialize H as a complete undirected graph over V ▷ Skeleton Recovery
for $i \leftarrow 0$ to $|V| - 2$ **do**
    while possible **do**
        Select any ordered pair of nodes u and v in H such that $u \in \mathrm{adj}_H(v)$ and $|\mathrm{adj}_H(u) \setminus \{v\}| \geq i$
        for each subset $S \subset \mathrm{adj}_H(u) \setminus \{v\}$ with $|S| = i$ **do**
            if $u \perp\!\!\!\perp_p v \mid S$ **then**
                Set $S_{uv} = S$
                Remove edge $u - v$ from H
            end if
        end for
    end while
end for ▷ v-structure Recovery
for each separator S_{uv} **do**
    if $u - w - v$ exists in H and $w \notin S_{uv}$ **then**
        Orient $u \rightarrow w \leftarrow v$ to form a v-structure
    end if
end for
Return H

2.2 Simulating the PC algorithm

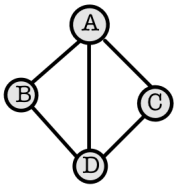
Example 1. Consider a dataset with variables A, B, C, and D. Let the following conditional independence tests be satisfied (calculated from the data):

1. $B \perp\!\!\!\perp_p C \mid A$
2. $A \perp\!\!\!\perp_p D \mid \{B, C\}$

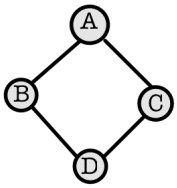
We initialize a graph H, which is the complete graph K_4.



After the first iteration ($i = 1$), the algorithm removes the edge $B - C$ due to $B \perp\!\!\!\perp_p C \mid A$. We set $S_{BC} = \{A\}$.



In the next iteration ($i = 2$), the edge $A - D$ is removed due to $A \perp\!\!\!\perp_p D \mid \{B, C\}$. We set $S_{AD} = \{B, C\}$ This yields the final skeleton.

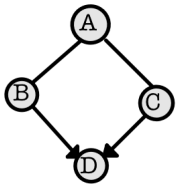


In the v-structure recovery phase, the algorithm orients edges to resolve directed relationships based on separation sets. We iterate over all the separation sets: S_{AD} and S_{BC}.

- S_{AD}: We first try to find a common neighbor for A, D that is not in the separation set S_{AD}. No such neighbor exists as $V/(\{A, D\} \cup S_{AD}) = \emptyset$

- S_{BC}: Observe that D neighbors B, C and $D \in V/(\{B, C\} \cup S_{BC})$ Thus, we orient $B - D - C$ as $B \rightarrow D \leftarrow C$

After resolving all v-structures and ensuring the graph remains acyclic, the resulting partially directed acyclic graph (PDAG) is:



This result also begs the question of how to convert the PDAG returned by PC, also called a *pattern* (CITE), to a DAG. This will be explored in the coming section.

2.3 Dealing with undirected edges

3 Score-Based Algorithms

The score-based approach first defines a criterion to evaluate how well the Bayesian network fits the data, then searches over the space of DAGs for a structure achieving the maximal score. The score-based approach is essentially a search problem that consists of two parts: the definition of a score metric and the search algorithm.

3.1 Score Metrics

$$Score(G; D) = LL(G; D) - \phi(|D|)|G|$$

Here, $LL(G; D)$ refers to the log-likelihood of the data under graph G. $\phi(|D|)|G|$ is a regularization term, which penalizes more complex models - favouring simpler ones.

3.2 The Chow-Liu Algorithm

The Chow-Liu Algorithm is a specific type of score-based approach that learns a Bayesian network structure by finding the maximum-likelihood tree-structured graph. In this algorithm, each node has exactly one parent (except for the root node), forming a directed tree. It achieves this by maximizing the mutual information between pairs of variables.

The algorithm proceeds as follows:

1. Compute the **mutual information** (which is the "score" for the Chow-Liu Algorithm) for all pairs of variables X, U, which quantifies the amount of shared information between them:

$$MI(X,U) = \sum_{x,u} \hat{p}(x,u) \log \frac{\hat{p}(x,u)}{\hat{p}(x)\hat{p}(u)}.$$

This measures how much information U provides about X.

Mutual information captures how much two variables (e.g., songs in a playlist) depend on each other. Variables with higher mutual information are more likely to have strong connections. For instance, if X and U are independent,

$$\forall x, u \; \hat{p}(x,u) = \hat{p}(x)\hat{p}(u)$$

$$\implies MI(X,U) = \sum_{x,u} \hat{p}(x,u) \log \frac{\hat{p}(x)\hat{p}(u)}{\hat{p}(x)\hat{p}(u)}. = \sum_{x,u} \hat{p}(x,u) \log 1 = 0$$

Also observe that, $\hat{p}(x,u)$ represents the empirical distribution over x and u,

$$\hat{p}(x,u) = \frac{N(x,u)}{N}$$

with $N(x,u)$ representing the number of data points with $X = x$ and $U = u$, and N being the total number of data points. Then,

$$Score(G;D) = LL(G;D) = |D| \sum_{(X,U)\in E} MI(X,U)$$

a proof of which is the appendix.

2. Construct a **complete graph** where each edge is weighted by the mutual information computed in Step 1.

3. Compute the **maximum-weight spanning tree (MST)** from the complete graph using a greedy algorithm such as Kruskal's or Prim's algorithm. The MST ensures that the resulting structure maximizes dependencies while avoiding cycles.

   Recall that a **spanning tree** of a graph G refers to a connected, undirected subgraph that contains all the vertices in G. A **MST** is simply the one with the maximum total weight among all spanning trees (where total weight is the sum of all edge weights). The weights in this scenario are given by the mutual information values calculated in step 1.

4. Select any node to be the **root variable**, and assign directions to the edges such that all arrows "radiate outward" from the root. This step transforms the undirected tree into a directed tree structure. The final step of assigning directions ensures a valid Bayesian network, where the tree structure naturally satisfies the Markov condition.

It can be observed that the Chow-Liu algorithm is more computationally efficient that PC, with a complexity of:

- $O(n^2)$ to compute mutual information for all pairs of n variables.
- $O(nlog(n))$ to compute the MST (e.g., using Kruskal's or Prim's algorithm).

Thus, the overall complexity is $O(n^2)$, which is significantly faster than algorithms that search the entire DAG space, as well as constraint-based algorithms like PC which compute a large amount of conditional independence tests.

3.3 Optimality of the Chow-Liu Algorithm

Theorem 2. The Chow-Liu algorithm is optimal.

Proof. The Chow-Liu algorithm works because it finds the tree structure that maximizes the likelihood of the data. Specifically, the likelihood score decomposes into mutual information and entropy terms, as follows:

$$\log P(\mathcal{D} \mid \theta^{ML}, G) = |\mathcal{D}| \sum_i MI_{\hat{p}}(X_i, X_{\mathrm{pa}(i)}) - |\mathcal{D}| \sum_i H_{\hat{p}}(X_i)$$

where:

- $MI_{\hat{p}}(X_i, X_{\mathrm{pa}(i)})$ is the mutual information between X_i and its parent $X_{\mathrm{pa}(i)}$ in the graph G.
- $H_{\hat{p}}(X_i)$ is the entropy of X_i.

The proof is this is given in **Appendix, Theorem 1**. The goal of any score-based algorithm is to maximize the "score" (log-likelihood) which is equivalent to maximizing the sum of the mutual information terms, as the entropy terms are independent of the dependency relationships. Thus,

$$\arg\max_G \log P(\mathcal{D} \mid \theta^{ML}, G) = \arg\max_G \sum_i MI(X_i, X_{pa(i)}).$$

This shows that the optimal tree structure is the one that maximizes the sum of mutual information values along its edges - making it a valid and optimal score metric. By computing the MST based on mutual information, the Chow-Liu algorithm ensures this optimality.

3.4 Simulating the Chow-Liu Algorithm

Example 2. Consider a dataset with variables A, B, C, and D. The mutual information between each pair of variables is calculated from the dataset.

$$\begin{aligned} &MI(A,B) = 0.32, \quad &&MI(B,C) = 0.17, \quad &&MI(C,D) = 0.02, \\ &MI(A,C) = 0.32, \quad &&MI(A,D) = 0.07, \quad &&MI(B,D) = 0.32. \end{aligned}$$

First, we construct a complete graph with edges weighted by mutual information.

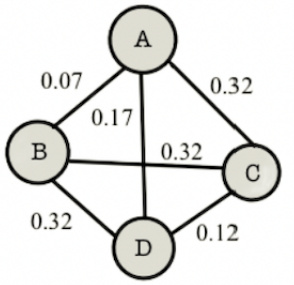


Figure 1: Fully Connected Graph with Mutual Information Edge Weights

Second, we use Prim/Kruskal's algorithms to find the Maximum Spanning Tree. Effectively, we choose the edges with the highest weights: $A - C$, $B - C$, and $B - D$, which add up to a total weight of 0.96.

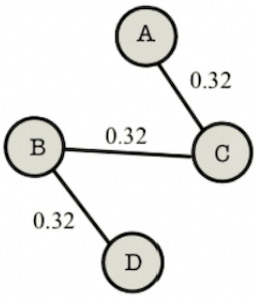


Figure 2: Maximal Spanning Tree arising from Figure 1

Lastly, we orient the tree edges by arbitrarily choosing a root and pointing the edges outwards.

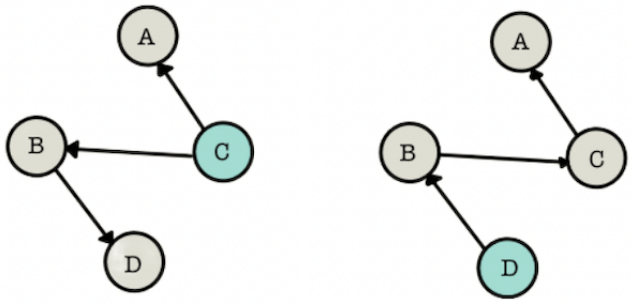


Figure 3: Choosing C as the root yields the graph on the left. Choosing D as the root yields the graph on the right.

As can be seen, the resulting graph is a tree that spans all the vertices and maximizes mutual information. Note that there are also more recent variants of the Chow-Liu algorithm which involve the potential creation of forests, as in the PC algorithm.

4 Applications with a Music Recommender System

This section will involve the application of the Peter-Clark and Chow-Liu algorithms on a dataset provided by Spotify (CITE), in order to produce the foundations for a music recommender system. Indeed, building a complete system would necessitate parameter learning to completely characterize the Bayesian Network - which is beyond the scope of this paper. The focus of this section, therefore, will be completely on the structure learning aspect of the recommender system.

4.1 Understanding the dataset

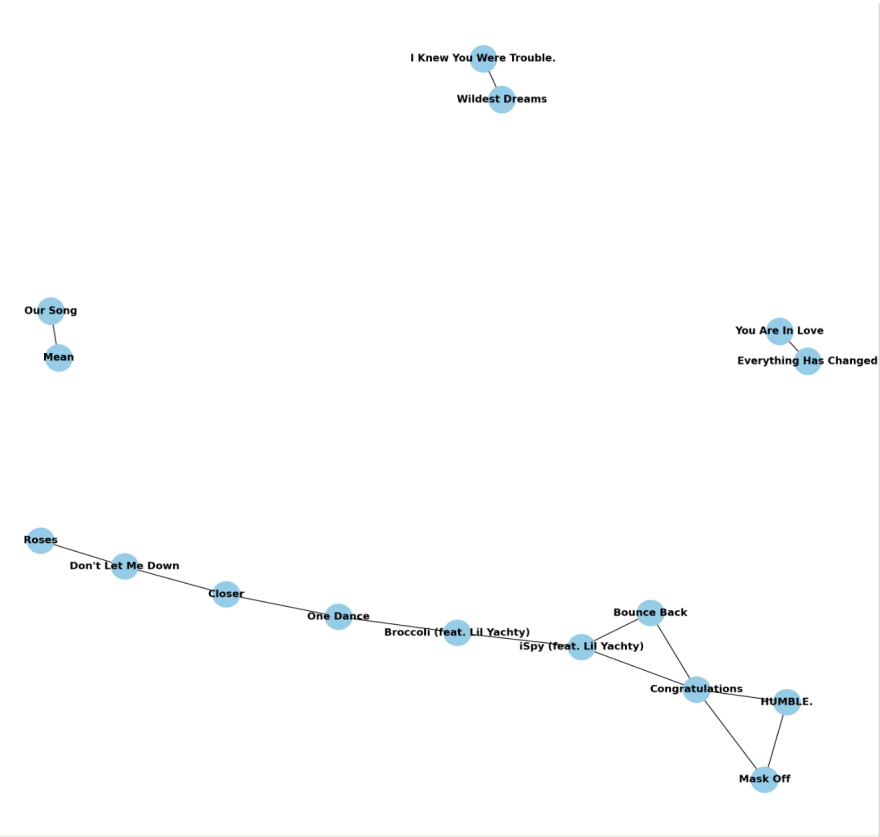
The entire dataset consists of 1 million playlists $P_1, .., P_N$, each of which contains a number of music tracks. Each track is identified with a unique resource identifier (uri), and metadata such as the track name, artist's name, etcetera. For the purpose of this analysis, I have significantly reduced the size of the dataset. First, we'll consider results across 1000 playlists, and then across 10000.

Further, 20 songs will be analyzed. 10 of these correspond to the top 10 most frequently appearing tracks across playlists. The other 10 are Taylor Swift songs which are included in the dataset. The expectation is for there to be a strong dependency relationship among Taylor Swift songs - after all, once you listen to your first Taylor Swift song, can you ever stop? (if you couldn't tell already, I love Taylor Swift) One would even expect some correlation among the top hits. To convert the creation of the recommender system to the general framework of a structure learning problem, we treat the songs $S_1, ..., S_{20}$ to be our boolean random variables (or attributes). S_i represents whether or not song i is in the given playlist, and thus $S_i \in \{0, 1\}$. Our data consists of the playlists (rows) $P_1, ..., P_N$ for $N = 1000$ and then $N = 10000$.

4.2 Results for $N = 1000$

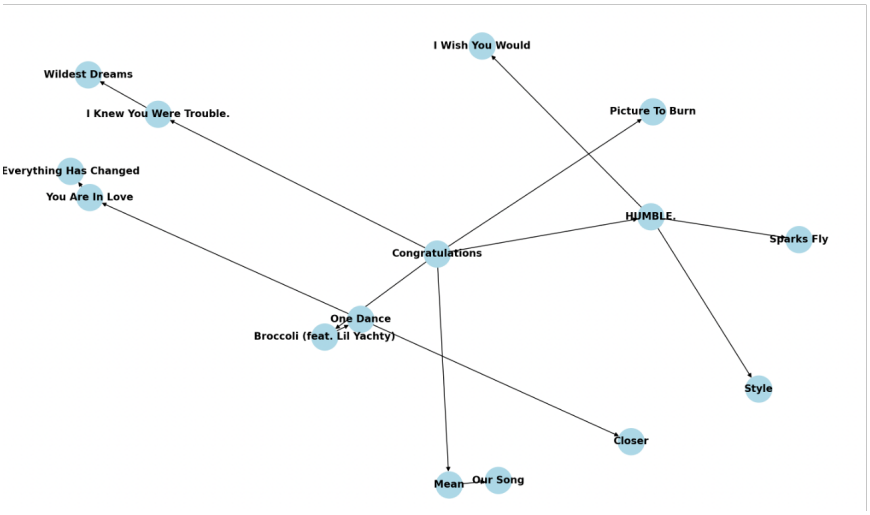
The pgmpy library in Python (CITE) proved to be incredibly useful for applying the PC and Chow-Liu algorithms. The code for applying the algorithm is provided in the appendix, and simply applies the required algorithm on a dataframe with columns as attributes (S_i) and rows as data points (P_i).

The resulting graph for the Peter-Clark algorithm is below.



As expected, the PC algorithm produces a forest with disjoint components of the graph. It is able to recognize a connection between Taylor Swift songs, however it doesn't cluster all her tracks in a strongly connected component. All "hit songs" seem to be strongly dependent on one another, which is an intuitive conclusion.

The graph yielded by the Chow-Liu algorithm follows.

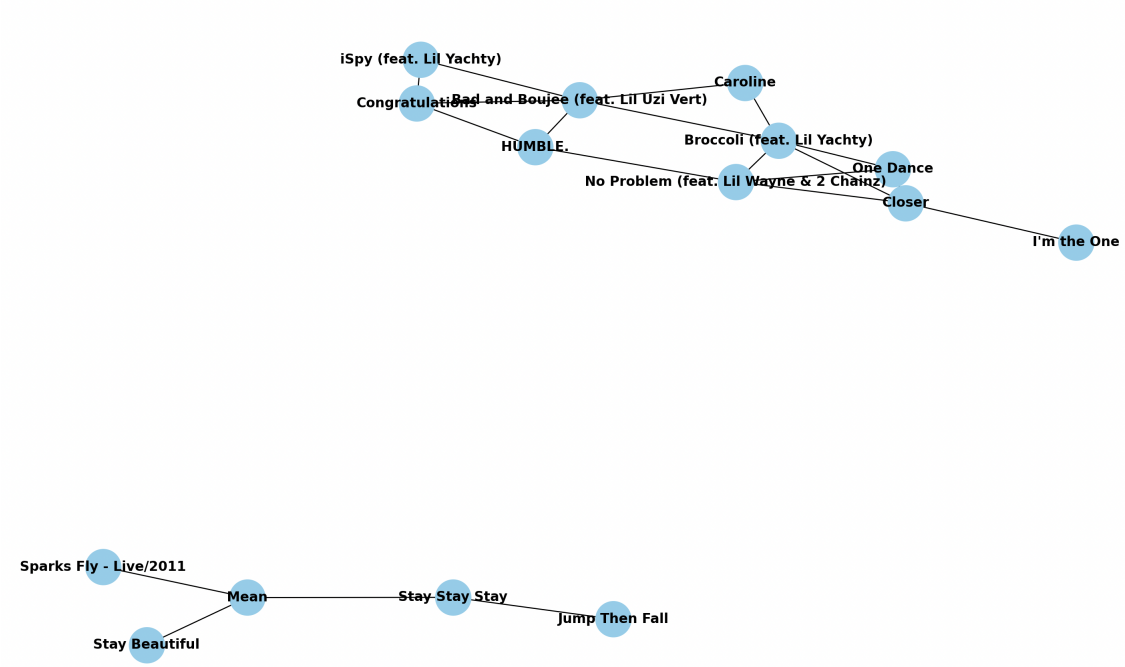


As expected, this particular implementation of the algorithm results in a tree structure (with each "song node" having only parent). Further, ignoring the edge directions, it is a **connected graph** (each node being "reachable" by any other). This is in stark contrast to the forest produced by Peter-Clark. Further, it also yields strong connections between Taylor Swift songs and similar to the Peter-Clark algorithm, it does so in pairs: "Wildest Dreams" and "I know you were Trouble" being a pair, "Everything Has Changed" and "You are In Love" being another, among others. A run time analysis yields the following results (with Chow-Liu being faster), but a bigger sample size is needed to exprolate more general conclusions.

```
PC: Time taken is 0.536 seconds
Chow-Liu: Time taken is 0.341 seconds
```

Results for N = 10000

The resulting graph for the Peter-Clark algorithm is below.



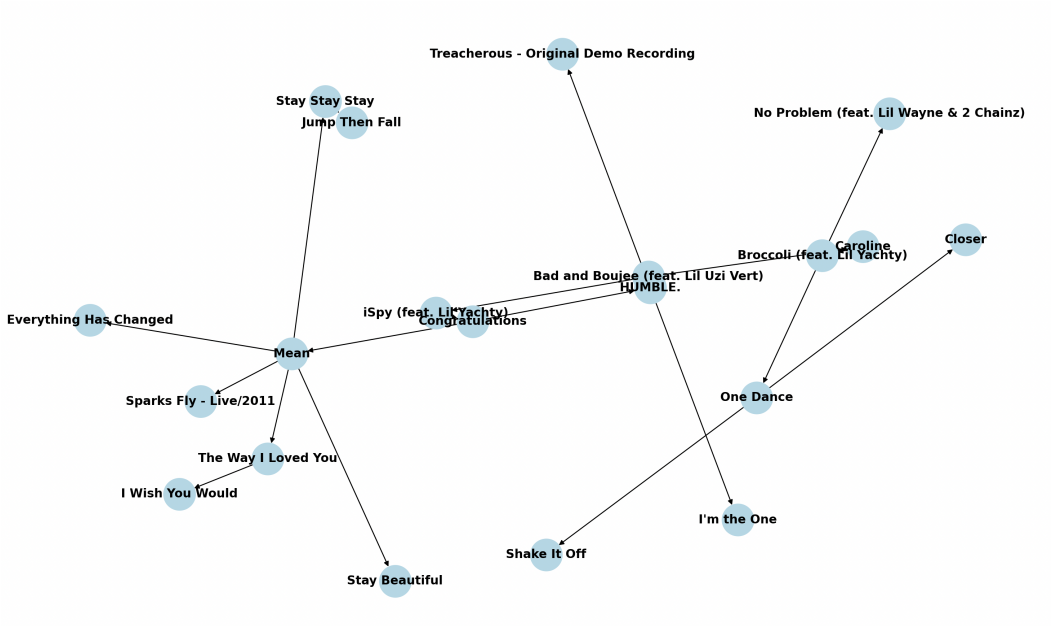
Again, a forest is observed. The results this time are clear - the PC algorithm has recognized the patterns and dependency relationships almost perfectly! All the Taylor Swift songs are in a connected component on the lower left, and all the "hit songs" are in their own connected component on the upper right. The actual results meet the expected ones, however there are two main caveats.

- The aforementioned issue of PC generating a PDAG in place of a DAG, although a possible solution for this was discussed in *section 2.3*
- The runtime greatly exceeds that of Chow-Liu - an expected result of the increasing number of calculations that must be performed to determine conditional independence.

The runtime comparison with the larger sample size is below. PC's runtime is more than 7 times greater than that of Chow-Liu.

```
PC: Time taken is 15.06 seconds
Chow-Liu: Time taken is 2.03 seconds
```

Chow-Liu's accuracy in determining dependency relationships may not be as sureshot, however.



Visually, "accuracy" in dependency detection would entail connections between all Taylor Swift songs (as these maximize mutual information). However, while a majority of her songs are indeed connected, some (like "Shake It Off") are further away from the bulk of the songs. These outliers indicate that it may not have been able to extrapolate these relationships as well as PC. Yet again, its relatively quicker runtime and edge directions provide more information about the structure of the data.

1 5 Conclusions

Appendix

Theorem 1. For the Chow-Liu Algorithm, the score metric (which is the likelihood of the data $\mathcal{D}$ given a tree-structured Bayesian network G), is equivalent to the sum of all mutual information terms in the tree structure. Equivalently,

$$Score(G; D) = LL(G; D) = |D| \sum_{(X,U)\in E} MI(X, U)$$

Proof. In a tree-structured Bayesian network G, each variable X_i depends on a single parent $X_{\text{pa}(i)}$, except for the root, which has no parent. The joint probability distribution over the variables $X_1, X_2, \ldots, X_n$ is:

$$P(X_1, X_2, \ldots, X_n \mid G) = \prod_{i=1}^{n} P(X_i \mid X_{\text{pa}(i)}).$$

Thus, for a dataset $\mathcal{D}$ consisting of $|\mathcal{D}|$ independent samples, we have the likelihood as

$$P(\mathcal{D} \mid G) = \prod_{d \in \mathcal{D}} P(X_1^{(d)}, X_2^{(d)}, \ldots, X_n^{(d)} \mid G).$$

Substituting the joint probability:

$$P(\mathcal{D} \mid G) = \prod_{d \in \mathcal{D}} \prod_{i=1}^{n} P(X_i^{(d)} \mid X_{\mathrm{pa}(i)}^{(d)}).$$

Taking the logarithm of the likelihood we turn the products into summations, yielding

$$\log P(\mathcal{D} \mid G) = \sum_{d \in \mathcal{D}} \sum_{i=1}^{n} \log P(X_i^{(d)} \mid X_{\mathrm{pa}(i)}^{(d)}).$$

The probabilities $P(X_i \mid X_{\mathrm{pa}(i)})$ are estimated using empirical distributions $\hat{p}(x_i, x_{\mathrm{pa}(i)})$ and $\hat{p}(x_{\mathrm{pa}(i)})$:

$$P(X_i \mid X_{\mathrm{pa}(i)}) = \frac{\hat{p}(x_i, x_{\mathrm{pa}(i)})}{\hat{p}(x_{\mathrm{pa}(i)})}.$$

Furthermore, we have

$$log\ P(X_i^{(d)} \mid X_{\mathrm{pa}(i)}^{(d)}) = log(\frac{\hat{p}(x_i, x_{\mathrm{pa}(i)})}{\hat{p}(x_{\mathrm{pa}(i)})})$$

where according to the definition of an **emperical distribution**,

$$\hat{p}(x_i, x_{\mathrm{pa}(i)}) = \frac{\mathrm{Count}(x_i, x_{\mathrm{pa}(i)})}{|\mathcal{D}|}, \quad \hat{p}(x_{\mathrm{pa}(i)}) = \frac{\mathrm{Count}(x_{\mathrm{pa}(i)})}{|\mathcal{D}|}.$$

$$\implies \frac{\mathrm{Count}(x_i, x_{\mathrm{pa}(i)})}{\mathrm{Count}(x_{\mathrm{pa}(i)})} = \frac{\hat{p}(x_i, x_{\mathrm{pa}(i)})}{\hat{p}(x_{\mathrm{pa}(i)})}.$$

Substituting this into the log-likelihood, we get:

$$\log P(\mathcal{D} \mid G) = \sum_{d \in \mathcal{D}} \sum_{i=1}^{n} \log \left(\frac{\mathrm{Count}(x_i, x_{\mathrm{pa}(i)})}{\mathrm{Count}(x_{\mathrm{pa}(i)})} \right).$$

Next, we rewrite the summation to consider unique values of $(x_i, x_{\mathrm{pa}(i)})$:

$$\log P(\mathcal{D} \mid G) = \sum_{i=1}^{n} \sum_{x_i, x_{\mathrm{pa}(i)}} \mathrm{Count}(x_i, x_{\mathrm{pa}(i)}) \log \left(\frac{\mathrm{Count}(x_i, x_{\mathrm{pa}(i)})}{\mathrm{Count}(x_{\mathrm{pa}(i)})} \right).$$

Substituting this back into the equation:

$$\log P(\mathcal{D} \mid G) = \sum_{i=1}^{n} \sum_{x_i, x_{\mathrm{pa}(i)}} |\mathcal{D}| \cdot \hat{p}(x_i, x_{\mathrm{pa}(i)}) \log \frac{\hat{p}(x_i, x_{\mathrm{pa}(i)})}{\hat{p}(x_{\mathrm{pa}(i)})}.$$

The term inside the summation is nothing but the **mutual information** $MI(X_i, X_{\text{pa}(i)})$ between X_i and its parent $X_{\text{pa}(i)}$:

$$MI(X_i, X_{\text{pa}(i)}) = \sum_{x_i, x_{\text{pa}(i)}} \hat{p}(x_i, x_{\text{pa}(i)}) \log \frac{\hat{p}(x_i, x_{\text{pa}(i)})}{\hat{p}(x_i)\hat{p}(x_{\text{pa}(i)})}.$$

Hence, substituting $MI(X_i, X_{\text{pa}(i)})$ into the log-likelihood:

$$\log P(\mathcal{D} \mid G) = |\mathcal{D}| \sum_{i=1}^{n} MI(X_i, X_{\text{pa}(i)}).$$

Lastly, given we have a tree structure, the sum of mutual information terms can equivalently be written over the edges E of the tree as

$$\log P(\mathcal{D} \mid G) = |\mathcal{D}| \sum_{(X,U) \in E} MI(X, U).$$

which thereby becomes the **Score Metric** for the Chow-Liu algorithm.