Shortest Learning Paths in Bayesian Networks - LazyLearner

Peer Schendel

January 5, 2022

1 Problem Description - Real World Problem

Our goal is to help students to catch up certain competences in an e-learning environment by minimizing the number of courses a student has to learn to pass a target course. For example a student moves to another city with a different curriculum and has to catch up a specific subject which can be learnt by an e-learning course. To assist the student in this matter we try to recommend a shortest learning path regarding e-learning courses.

2 Example

We construct a small example where a student on an example Bayesian network. A student wants to learn geometry and there are different courses that have a causal inference on passing the course geometry.

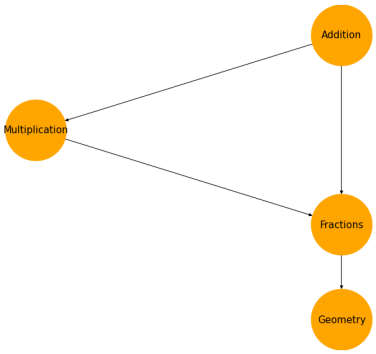

Figure 1: Example Bayesian network: Different competences that are needed to learn geometry

3 Preliminaries

3.1 Bayesian Network (BN) - Data Structure

We are using a Bayesian network as a given data structure for our problem. It is widely used for causal inference.

We can define a Bayesian network by the following properties[1]:

- Probabilistic graph model
- Directed acyclic graph (DAG)
- Edges correspond to a causal relationship between nodes

- Markov condition is fulfilled, that means the probability of one node only depends on their parents

We can calculate the joint probability by $P(x_1, ..., x_n) = \prod_i P(x_i|pa_i)$ where pa_i are the corresponding parents to the node x_i.

3.2 Minimal Cardinality Explanation

We define our decision function, whether a given instantiation of variables yields a probability higher than the threshold t.

$$f(x) := \begin{cases} 1 \ if \ P(target \ node = 1|x) \geq t \\ 0 \ else \end{cases}$$

The MC-explanation answers the question: "Which positive features of an instance x are responsible for the decision?"
MC-explanations are not necessarily unique but must have the identical number of 1-features. [2]

3.3 Conditional Probability Tables (CPT)

In Bayesian networks we can take the advantage of conditional independence that leads to a reduction of parameters. There is a smaller representation called conditional probability tables that represent the probability of each realization of a node's parents. We can use those CPTs per node as a mapping for our IP modelling.

4 Problem Description

Given: Bayesian network $G = (V, E, P)$, a target node $S \in V$ and a threshold $t \in [0, 1]$
Goal: Find a MC-explanation of positive decision function of target node or return infeasibility. The decision function has to return 1 for all positive nodes.

We can than use the topological ordering of the Bayesian network for our MC-explanation and construct a shortest path on a complete graph.

5 Complexity of the Problem

No proof was found in the literature about the complexity or the problem in general. For this reason, we content ourselves with a very simple and incomplete proof. We show that the problem lies in NP.
We can efficiently verify if a given instantiation of parent nodes yields a yes-instance that means $P(target \ node = yes \ / \ instantiated \ parents) \geq t$ by looking in our saved CPTs. We can construct an edge case of an instance by having $n - 1$ nodes as parents of 1 node:
The number of instances to check grow exponentially by 2^{n-1} in the worst case and thus

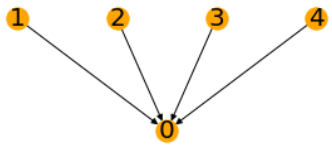

Figure 2: Edge case of $n - 1$ parents

to find a MC-explanation of such set yields that the complexity of the problem lies in NP.

6 Modelling as IP

We can formulate our problem as an IP formulation where we define t as the threshold with $t = 0.5$ for simplification, n the number of nodes in the graph G and S the target node in G. We want to solve a minimax problem by minimize the number of positive nodes and maximize the probability of the target node.

$$\min \quad z = -P(S = 1|Y_{s-}) + \sum_{i=1}^{n} y_i \tag{1}$$

$$\text{subject to:} \quad P(S = 1|Y_{s-}) \geq t \tag{2}$$

$$P(Y_i = 1|Y_{i-}) \geq t \cdot y_i \quad \forall i = 1, n \tag{3}$$

$$y_i \in \{0, 1\} \quad \forall i = 1, n \tag{4}$$

We define $Y_{i-} := \{Y_1 = y_1, ..., Y_{i-1} = y_{i-1}, Y_{i+1} = y_{i+1}, ..., Y_n = y_n\}$

7 Solver and preprocessing steps

For this project we used the GEKKO[P2] solver which has an open-source license. We did the following preprocessing steps:

1. Get all parents recursively
2. Create CPT for every node by looping through all realizations of parents
3. Create binary variable for each node in set of all parents (1.)
4. For each node enter the recursive constraints for each realization of parents
5. Enter recursive constraints for target node
6. Add maximization of probability of target node

We had to implement a constraint for each realization of parents because we cannot use any functions besides internal functions of GEKKO. That is why we calculate all CPTs and then add a constraint for each realization and check whether it is identical to the binary variables and if so then we check if the node is taken and check the threshold condition.
So there are some extra constraints built in in contrast to the Integer Program.
The preprocessing runs with many for loops which may be vectorized for a speed up in the program.

8 How the Test instances where generated

To build test instances we used the package pybbn [P1] where we implemented small instances with known outputs and joined them together by an edge and run the program.Therefore we changed the CPTs of the successor node and relabeled all nodes of the appended graph.

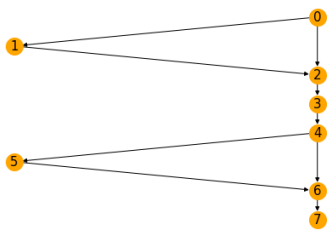

Figure 3: Edge case of $n - 1$ parents

9 Computational Results

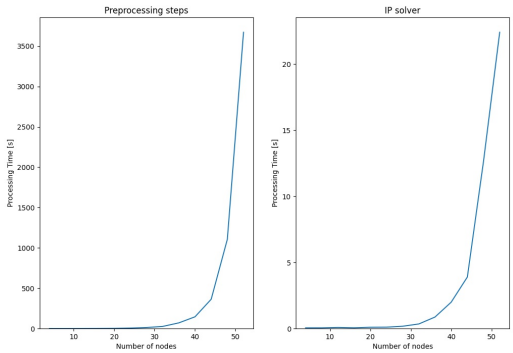

Figure 4: Processing time of per increasing number of nodes generated as discussed in 8

We can see that the computational time has an elbow point at around 30-40 nodes. After we appended 13 test instances together the preprocessing time takes already one hour while the solver already is higher than 20s. We can also see that the solver stays under 1s for smaller instances unfortunately we have to preprocess and enter every condition beforehand that we cannot simply use the solver.

10 Conclusion

We can see that the calculation for larger instances takes long time. There is no real-time application of our algorithm but the IP formulation did solves the recursive condition by only adding a constraint. For a practical use the program has to be precalculated and saved as another data structure for example ordered binary decision diagrams which can solve the MC-explanation in linear time.[3]

11 Sources

Code:

- https://github.com/schpeer92/Optimal-Learning-Path-Recommender

Package sources:

- https://py-bbn.readthedocs.io/ [P1]
- https://gekko.readthedocs.io/en/latest/ [P2]
- Pearl, J., Russel, S. (2000, November). BAYESIAN NETWORKS. Cambridge; MIT Press[1]
- Boutilier, C., Friedman, N., Goldszmidt, M., Koller, D. (1996). Context-Specific Independence in Bayesian Networks. Proceedings of the Twelfth Conference on Uncertainty in Artificial Intelligence (UAI1996). [2]
- Shih, A., Choi, A., Darwiche, A. (2018, May 9). A Symbolic Approach to Explaining Bayesian Network Classifiers. [3]