Probabilistic Graphical Models (CSE 516)
Project Report

Kshitij Shekhar (AU20202165)

November 12, 2022

School of Engineering and Applied Science (SEAS), Ahmedabad University

I. Introduction

A. Background

Globally, cardiovascular disease is the leading cause of death. Data mining from the medical field can aid in finding useful information from available data. Cardiovascular disorders (CVDs) are caused by heart and blood vessel dysfunction. To determine which individuals are at a higher risk of a heart attack based on several characteristics, Bayesian networks are used. These networks are a type of probabilistic graphical model and are represented as directed acyclic graphs with nodes and edges.

B. Motivation

CVDs are a primary cause of death globally, particularly in developed, high-income countries. Many factors contributing to CVD are complex for a layperson to understand. Advances in AI, including probabilistic graphical models, have helped democratize data and its applications. While classic machine learning models like SVMs and Random Forests can predict heart disease, they often lack explainability. Deep Learning models, though often achieving state-of-the-art results with large datasets, function as "black boxes." Probabilistic models like Bayesian Networks, however, are valued for their interpretability. This project focuses on identifying relationships between 14 features from a dataset and performing inference. A Bayesian Network can illustrate conditional dependencies and independencies among these features.

C. Contribution

The authors of the referenced paper implemented a Bayes Net in R using the `bnlearn` library.This project aims to implement the same in Python using `pgmpy` and attempt to match or improve upon the performance metrics achieved by the original authors.

II. Methodology

A. Analytical Overview/Product Details

The product is a web application built with the Python framework Anvil. It allows users to input patient features through a simple interface with label fields and corresponding text boxes. A "PREDICT" button triggers inference on the input data via a server connection with a Colab notebook, outputting a prediction (1 for positive heart disease diagnosis, 0 for negative).

B. Mathematical Analysis/Working of the Product/Framework

This project involves constructing a Bayesian Network to predict cardiovascular disease onset based on patient-entered features. The model was trained on the Cleveland heart disease dataset from the UCI Machine Learning Repository. Structure learning was performed on 80

A Bayesian Network is defined by its structure (a Directed Acyclic Graph - DAG) and its parameters (the CPDs associated with each node). The joint probability distribution over a set of random variables $X_1, X_2, \ldots, X_n$ can be factorized according to the network structure:

$$P(X_1, X_2, \ldots, X_n) = \prod_{i=1}^{n} P(X_i|\text{Pa}(X_i))$$

where $\text{Pa}(X_i)$ denotes the set of parent nodes of X_i in the graph. This factorization highlights the conditional independence assumptions encoded in the network: each variable X_i is conditionally independent of its non-descendants, given its parents.

For each node X_i, a CPD, $P(X_i|\text{Pa}(X_i))$, quantifies the probability of X_i taking on each of its possible values for every combination of its parents' values. These are often represented as tables (TabularCPDs) for discrete variables.

Structure Learning

The project employed score-based structural learning algorithms. These algorithms search for a network structure that maximizes a chosen scoring function. The report mentions using `HillClimbSearch` with `K2Score` (and `BicScore` is mentioned in code imports).

- **K2 Score:** A Bayesian score that assumes a uniform prior over the space of possible structures and parameters. It is computationally efficient but requires an ordering of variables. `pgmpy`'s `K2Score` was used.
- **BIC (Bayesian Information Criterion) Score:** An information-theoretic score that penalizes model complexity. It is defined as:

  $$\text{BIC} = \log(L) - \frac{k}{2}\log(N)$$

  where L is the maximized value of the likelihood function for the model, N is the number of data points, and k is the number of parameters in the model. The goal is to maximize this score.

Parameter Learning

After determining the structure, the CPDs were learned from the training data. The methods used include:

- **Maximum Likelihood Estimation (MLE):** This approach finds the parameter values that maximize the likelihood of observing the given data. For a variable X_i with parents $\text{Pa}(X_i)$, the MLE estimate for $P(X_i = x_i|\text{Pa}(X_i) = \text{pa}_i)$ is given by:

  $$\hat{\theta}^{\text{MLE}}_{x_i|\text{pa}_i} = \frac{N(X_i = x_i, \text{Pa}(X_i) = \text{pa}_i)}{N(\text{Pa}(X_i) = \text{pa}_i)}$$

  where $N(\cdot)$ is the count of occurrences in the dataset.
- **Bayesian Estimation:** This method starts with a prior probability distribution for the parameters and updates it based on the observed data to obtain a posterior distribution. The estimate is often the mean of this posterior. The project used `BayesianEstimator` with `prior_type="BDeu"`. For example, using a Dirichlet prior (common for categorical distributions), the estimate for $P(X_i = x_i|\text{Pa}(X_i) = \text{pa}_i)$ with BDeu (Bayesian Dirichlet equivalent uniform) prior is:

  $$\hat{\theta}^{\text{Bayes}}_{x_i|\text{pa}_i} = \frac{N(X_i = x_i, \text{Pa}(X_i) = \text{pa}_i) + \alpha_{x_i|\text{pa}_i}}{N(\text{Pa}(X_i) = \text{pa}_i) + \sum_{x'_i} \alpha_{x'_i|\text{pa}_i}}$$

  where α represents the hyperparameters from the Dirichlet prior (pseudocounts). The `BDeu` prior often sets $\alpha_{ijk} = \frac{\text{ISS}}{r_i q_i}$ where ISS is the imaginary sample size, r_i is the cardinality of X_i, and q_i is the product of cardinalities of $\text{Pa}(X_i)$.
- **Expectation Maximization (EM):** Used when there is missing data or hidden variables. The dataset reportedly had no missing values after preprocessing.

Inference

Once the model is learned, inference can be performed. The project uses `VariableElimination` to compute probabilities. Specifically, `map_query` is used, which finds the Most A Posteriori (MAP) assignment for the query variable(s) given evidence. For query variables Q and evidence $E = e$:

$$\text{MAP}(Q|E = e) = \arg\max_q P(Q = q|E = e)$$

The model was tested using accuracy, precision, and recall.

- **Accuracy:** $\frac{\text{True Positives+True Negatives}}{\text{True Positives+True Negatives+False Positives+False Negatives}}$
- **Precision:** $\frac{\text{True Positives}}{\text{True Positives+False Positives}}$
- **Recall (Sensitivity):** $\frac{\text{True Positives}}{\text{True Positives+False Negatives}}$

There's often a trade-off between precision and recall.

The authors of the referenced paper used `bnlearn` in R, and differences in implementation between `bnlearn` and `pgmpy` might lead to variations in model performance.

The chain rule for probabilities is mentioned in the report:

$$P(X_1 = x_1, \ldots, X_n = x_n) = \prod_{v=1}^{n} P(X_v = x_v|X_{v+1} = x_{v+1}, \ldots, X_n = x_n)$$

In the context of a Bayesian Network, this simplifies due to conditional independencies:

$$P(X_1 = x_1, \ldots, X_n = x_n) = \prod_{v=1}^{n} P(X_v = x_v|X_j = x_j \text{ for each } X_j \text{ that is a parent of } X_v)$$

This is the same as the previously mentioned factorization $P(X_1, \ldots, X_n) = \prod_{i=1}^{n} P(X_i|\text{Pa}(X_i))$.

Nodes not connected by edges are conditionally independent of one another, given certain conditions related to paths between them (d-separation). The report mentions that the path between any two nodes in a BN is said to be blocked if there is at least 1 such condition.

III. Codes

(The code section is detailed in the provided PDF. Key libraries imported include `pgmpy.models.BayesianNetwork`, `pgmpy.factors.discrete.TabularCPD`, `pgmpy.estimators` like `HillClimbSearch`, `BicScore`, `PC`, `K2Score`, `BayesianEstimator`, `MaximumLikelihoodEstimator`, `ExpectationMaximization`, and `pgmpy.inference.VariableElim`

IV. Results and Inferences

- **Reproduced Model (from reference paper, tested on the test set):**
  - Accuracy: 83.33%
  - Precision: 82.14%
  - Recall: 82.14%
- **Project's Bayesian Network (structure learned on the training set, parameter learning with Bayesian Estimator on the training set, tested on test set):**
  - Accuracy: 81.67%
  - Precision: 79.31%
  - Recall: 82.14%

Maximum Likelihood Estimator and Expectation Maximization yielded similar performance to the Bayesian Estimator.

V. Conclusion

The project utilized various Bayesian network methods from the `pgmpy` library to reproduce research and learn a proprietary Bayesian network for heart disease prediction.

VI. References

- Muibideen, Prasad. (2020, December 18). Arxiv. A FAST ALGORITHM FOR HEART DISEASE PREDICTION USING BAYESIAN NETWORK MODEL. Retrieved November 12, 2022, from `https://arxiv.org/pdf/2012.09429.pdf`
- Ankan. (2013). pgmpy 0.1.19 documentation. Pgmpy 0.1.19 Documentation. Retrieved November 12, 2022, from `https://pgmpy.org/`
- (n.d.). Turning a Google Colab Notebook into a Web App. Anvil. Retrieved November 12, 2022, from `https://anvil.works/learn/tutorials/google-colab-to-web-app`