Inference and Imputes in Bayesian Networks

Cyril Bousmar - 63401800
Louvain School of Engineering
cyril.bousmar@uclouvain.be

Group 23

Mohamed-Anass Gallass - 02652201
Louvain School of Engineering
anass.gallass@student.uclouvain.be

I. Introduction

In the context of the course LINFO2364: Mining Patterns in Data, we were tasked with implementing exact inference and learning techniques for Bayesian Networks. These probabilistic graphical models offer a compact and interpretable way to represent complex distributions over high-dimensional spaces using *Directed Acyclic Graphs (DAGs)* and *Conditional Probability Tables (CPTs*. The project involves reasoning under uncertainty, managing incomplete data, and designing scalable learning pipelines for real-world networks.

This report is organised as follows : Section II introduces the theoretical foundation of Bayesian Networks and inference strategies. Section III details our implementation of exact inference for marginal and joint probability queries. Sections IV and V explain respectively the *CPT* parameter estimation and the structure learning strategy. Section VI presents the evaluation pipeline we implemented on missing data prediction. Section VII reports the experimental results. Finally, section **??** discusses the limitations of our work and some perspectives for future work.

II. Bayesian Network Fundamentals

A Bayesian Network is a probabilistic graphical model representing a set of random variables and their conditional dependencies via a *Directed Acyclic Graph (DAG)*. Each node in the graph corresponds to a discrete random variable and each edge encodes a direct dependency. The joint probability distribution over all variables is factorized using the chain rule:

$$P(X_1, \ldots, X_n) = \prod_{i=1}^{n} P(X_i \mid \text{Parents}(X_i)) \tag{1}$$

Bayesian Networks enable compact representations of complex distributions and support reasoning under uncertainty. Given evidence on some variables, inference consists of computing the marginal or conditional probabilities of others.

This project employs exact inference methods (such as enumerating all possible combinations of target variable values) and then focuses on structure learning and parameter estimation for general Bayesian Networks.

III. Exact Inference for Missing Value Imputation

To perform exact inference in Bayesian Networks, we implemented an enumeration-based approach that computes the posterior probabilities $P(Y|X)$ for either one or two query variables which covers the missing value imputation scenario. Unlike traditional message-passing algorithms that require tree-structured networks, our approach handles arbitrary DAG structures through direct computation of joint probabilities.

For a single missing variable Y given evidence X, we compute:

$$\begin{aligned} P(Y = y \mid X = x) &= \frac{P(Y = y, X = x)}{P(X = x)} \\ &= \frac{P(Y = y, X = x)}{\sum_{y'} P(Y = y', X = x)} \end{aligned} \tag{2}$$

For two missing variables Y_1 and Y_2, we compute the joint distribution:

$$P(Y_1 = y_1, Y_2 = y_2 \mid X = x) = \frac{P(Y_1 = y_1, Y_2 = y_2, X = x)}{P(X = x)} \tag{3}$$

$P(Y_1, Y_2, X)$ is calculated by enumerating all possible assignments and using the network's factorization.

In our implementation, we tried to include some optimizations : all probability calculations are performed in logarithmic space to avoid numerical underflow with small probabilities. We also use the log-sum-exp trick for numerical stability.

IV. Parameter Learning

For the parameter learning step, we implemented the *Maximum Likelihood Estimation* with Laplace smoothing. We use the following formula :

$$\begin{aligned} &P(X_i = x_i \mid Pa(X_i) = pa(X_i)) \\ &= \frac{N(X_i = x_i, Pa(X_i) = pa(X_i)) + \alpha}{N(Pa(X_i) = pa(X_i)) + \alpha \cdot |X_i|} \end{aligned} \tag{4}$$

where we calculate the probability of a variable X_i taking a value x_i given that its parent $Pa(X_i)$ take on the $pa(X_i)$ with

- $N(X_i = x_i, Pa(X_i) = pa(X_i))$ is the count of instances where $X_i = x_i$ and $Pa(X_i) = pa(X_i)$
- $N(Pa(X_i) = pa(X_i))$ is the count of instances where $Pa(X_i) = pa(X_i)$
- α is the Laplace smoothing parameter (default = 1.0)
- $|X_i|$ is the cardinality of variable X_i (number of possible values)

We tried different values of α (0.01, 0.1, 0.5, 1.0, 2.0, 5.0) to find the optimal smoothing for each dataset. The smoothing value is important to prevent zero probabilities in sparse datasets.

In our implementation, we also use Pandas groupby operations for counting making it scalable to large datasets. For

variables with no parents, we compute simple frequency-based probabilities with appropriate smoothing.

After all that, the learned parameters are stored in the Bayesian Network's CPT structure ready for inference tasks.

V. Structure Learning

For the structure learning step of the project, we needed to discover the optimal DAG that best represents the dependencies in the data. To do that, we used the *hill climbing algorithm*. The approach consists of:

1) **Initialization**: Start with an empty graph (no edges).
2) **Hill Climbing**: Iteratively improve the structure by selecting the best local operation that increases the score.
3) **Tabu List**: Maintain a list of recently performed operations to avoid cycles.
4) **Random Restarts**: Perform multiple searches from different initial states.

In our code, we implemented three types of operations:

- **Edge Addition**: Add a directed edge between two variables $(X_i \rightarrow X_j)$.
- **Edge Removal**: Remove an existing edge between two variables.
- **Edge Reversal**: Reverse the direction of an existing edge $(X_i \rightarrow X_j)$ to $(X_j \rightarrow X_i)$.

Of course, each operation is subject to constraints. The resulting graph must remain acyclic (checked using DFS). A variable's parents cannot exceed the specified maximum (controlled by the $max\_parents$ parameter). Operations that are in the tabu list are skipped.

We use the Bayesian Information Criterion (BIC) to score candidate networks:

$$BIC(G, D) = LL(G, D) - \frac{k}{2} \log(N) \tag{5}$$

$$LL(G, D) = \sum_{i=1}^{N} \log P(x^{(i)}) \tag{6}$$

where:

- $LL(G, D)$ is the log-likelihood of the data given the network
- k is the number of parameters in the network
- N is the number of samples in the dataset
- $P(x^{(i)})$ is the probability of the i-th instance according to the network.

The BIC score includes a penalty term for model complexity, which helps prevent overfitting by favoring simpler networks.

In our implementation, we tried to includes some optimizations. The hill climbing process stops when no operation improves the score. We also test several different configurations with varying max parents and alpha parameters.

VI. Evaluation Strategy

A. Datasets Structure and Missing Data

We evaluate our models on six benchmark datasets of varying size and complexity, presented in Table I. Split ratio between training and testing set is 4/1. Each dataset has complete and artificially corrupted testing set versions. The latter has missing values randomly introduced.

TABLE I: Dataset statistics used for inference evaluation

Dataset	#Samples	#Variables	#Missing	Missing Rate
alarm	10,000	37	2001	2.70%
andes	10,000	223	1975	0.44%
asia	10,000	8	2074	12.96%
sachs	10,000	11	1970	8.95%
sprinkler	1,000	4	206	25.75%
water	10,000	32	2037	3.18%

They cover a wide spectrum:

- **Small sets** like *sprinkler* challenge robustness under sparse structure but high noise.
- **Large sets** such as *andes* test inference scalability with low uncertainty.
- **Balanced sets** like *alarm* and *water* allow realistic assessment of both accuracy and efficiency.

Our evaluation pipeline follows these steps:

1) Train a Bayesian Network on the complete training data.
2) Impute missing values in the test dataset.
3) Compare imputed values against ground truth.
4) Record metrics and structural characteristics.

We implemented an experiment tracking system that generates CSV files with detailed information about each experiment.

B. Setup

All experiments were run on a MacBook Air M1 with 8 CPU cores and 16GB of RAM.

VII. Results and Analysis

Figure 2 and Table II summarizes the performance of our Bayesian Network implementation across the benchmark datasets, comparing the learned structures with expert-designed networks.

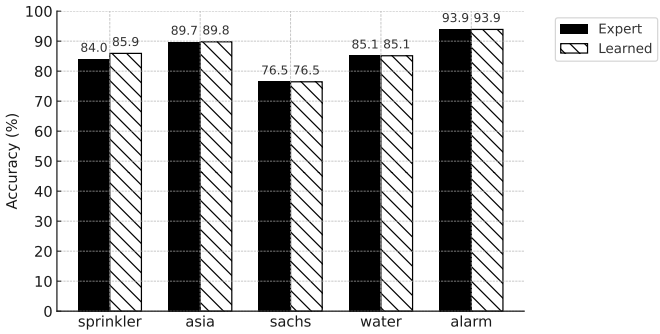

Fig. 2: Imputation accuracy : expert vs learned structures

TABLE II: Accuracy of learned vs. expert networks on missing value imputation

Dataset	Complete	Learned	Difference
sprinkler	83.98%	85.92%	+1.94%
asia	89.68%	89.78%	+0.10%
sachs	76.45%	76.45%	0.00%
water	85.08%	85.13%	+0.05%
alarm	93.90%	93.90%	0.00%

The learned networks achieved comparable accuracy than expert-designed networks across all datasets, and get even better results for the smaller sets (sprinkler and asia).

We observed that varying the `alpha` value in our the hyperparameters had no significant effect on accuracy for these datasets.

The training time varied significantly across datasets:

- Small datasets (sprinkler, asia): 1-250 seconds
- Medium datasets (sachs): 900 seconds
- Large datasets (water, alarm): 34,000-42,500 seconds

While most datasets were processed in a reasonable time-frame with `max_iters` set to at most 20, the *andes* dataset did not complete due to its high dimensionality. This highlights the exponential scaling of exact structure learning and the need for more efficient or approximate optimization methods for large networks.

VIII. Conclusion

In this project, we implemented a comprehensive Bayesian Network framework for exact inference and learning, with a focus on missing value imputation. Our solution included parameter learning with Laplace smoothing and structure learning using enhanced hill climbing with tabu search.

The experimental results demonstrated that our approach achieves good performance. The implementation successfully balanced model complexity and fit, avoiding overfitting while capturing essential dependencies in the data. The learned structure matched the complete expert-designed networks provided in terms of imputation accuracy.

However, the scalability remains a challenge for higher dimensionality datasets, especially here for *andes*. Future work should explore solving this issue.

Overall, our implementation provides solid foundation for probabilistic modelling and missing data imputation.

References

[1] J. Pearl. Probabilistic Reasoning. In *Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann, 1988.