Causal Models: Fitting

Alessio Zanga[1,*]

October 17, 2025

[1]Models and Algorithms for Data and Text Mining (MADLab), University of Milano-Bicocca, Milan, Italy

[*]Corresponding author: alessio.zanga@unimib.it

Introduction

Introduction to Bayesian Networks

- Bayesian Networks are probabilistic graphical models that represent a set of variables and their conditional dependencies via a Directed Acyclic Graph (DAG).
- Today, we will use the `pgmpy` library to work with the Alarm Bayesian Network.
- We will load this network from a BIF file using `pgmpy`'s `BIFReader`.

Interacting with the Model

- **Structural queries:**
  - We can query the network to find the parents, children, and Markov Blanket of specific nodes.
  - This helps in understanding the relationships and dependencies between nodes.
- **Independence queries:**
  - D-separation is a criterion for deciding whether a set X of nodes is independent of another set Y of nodes given a third set Z.
  - We will test d-separation for pairs of variables to understand their independencies.

Sampling from the Bayesian Network

- Bayesian networks can be used to generate synthetic data.
- We will use forward sampling to generate data from the Asia network.
- This data can be used for further analysis or testing.

Conditional Probability Distributions and Tables (CPDs & CPTs)

- A **Conditional Probability Distribution** defines the probability of a variable given its parents in the graph.
- For a variable X with parents $Pa(X)$, the CPD is $P(X|Pa(X))$.
- CPDs are essential for defining the joint:

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

- For categorical values we have conditional probabilty tables (CPTs).

Bayesian Network Interchange Format (BIF) (1/3)

- The basic unit of information is **a block**:
  - a piece of text which starts with **a keyword** ...
  - and ends with the end of an attribute list.
- Comments are allowed between blocks.

```
network asia {
   property version 1.1;
   property author nobody;
}
```

Bayesian Network Interchange Format (BIF) (2/3)

```
variable smoke {
  type discrete [ 2 ] { yes, no };
}
...
probability ( smoke ) {
  table 0.5, 0.5;
}
```

Bayesian Network Interchange Format (BIF) (3/3)

```
variable dysp {
  type discrete [ 2 ] { yes, no };
}
...
probability ( dysp | bronc, either ) {
  (yes, yes) 0.9, 0.1;
  (no, yes) 0.7, 0.3;
  (yes, no) 0.8, 0.2;
  (no, no) 0.1, 0.9;
}
```

Parameter Learning

Maximum Likelihood Estimator for a Categorical Distribution (1/3)

- **Problem:**
  - Let $X_1, \ldots, X_n$ be i.i.d. draws from a categorical distribution with K outcomes.
  - Parameters: $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_K)$, where $\theta_k = P(X = k)$ and $\sum_{k=1}^{K} \theta_k = 1$.
- **Likelihood:**

$$L(\boldsymbol{\theta}) = \prod_{i=1}^{n} \prod_{k=1}^{K} \theta_k^{\mathbb{1}\{X_i = k\}} = \prod_{k=1}^{K} \theta_k^{n_k},$$

  where n_k is the count of observations in category k.
- **Log-likelihood:**

$$\ell(\boldsymbol{\theta}) = \sum_{k=1}^{K} n_k \log \theta_k, \quad \text{subject to } \sum_{k=1}^{K} \theta_k = 1.$$

Maximum Likelihood Estimator for a Categorical Distribution (2/3)

- **a) Differentiate:**

$$\frac{\partial \ell}{\partial \theta_k} = \frac{n_k}{\theta_k} = c \quad \Rightarrow \quad \theta_k = \frac{n_k}{c}.$$

- **b) Normalize:**

$$\sum_{k=1}^{K} \theta_k = 1 \quad \Rightarrow \quad c = \sum_{k=1}^{K} n_k = n.$$

- The MLE is given by the **frequency of the categories**:

$$\boxed{\hat{\theta}_k = \frac{n_k}{n}}$$

Maximum Likelihood Estimator for a Categorical Distribution (3/3)

- The MLE is given by the **frequency of the categories**:

$$\boxed{\hat{\theta}_k = \frac{n_k}{n}}$$

- Dataset

	rain
1	yes
2	no
3	no

- Counts

rain	no	yes	total
	2	1	3

- Probability

rain	no	yes	total
	67%	33%	100%

Bayesian Estimator for a Categorical Distribution (1/3)

- **Problem:**
  - Let $X_1, \dots, X_n$ be i.i.d. draws from a categorical distribution with K outcomes.
  - Parameters: $\boldsymbol{\theta} = (\theta_1, \dots, \theta_K)$, where $\theta_k = P(X = k)$ and $\sum_{k=1}^K \theta_k = 1$.
- **Prior:**

$$\boldsymbol{\theta} \sim \mathrm{Dir}(\alpha_1, \dots, \alpha_K)$$

- **Posterior:**

$$\boldsymbol{\theta} \mid \mathbf{n} \sim \mathrm{Dir}(\alpha_1 + n_1, \dots, \alpha_K + n_K)$$

- **Posterior Mean:**

$$\mathbb{E}[\theta_k \mid \mathbf{n}] = \frac{n_k + \alpha_k}{n + \sum_{i=1}^K \alpha_i}$$

Bayesian Estimator for a Categorical Distribution (2/3)

- **Maximum A Posteriori (MAP):**

$$\boxed{\hat{\theta}_k = \frac{n_k + \alpha_k - 1}{n + \sum_i \alpha_i - K}}$$

- **Special cases:**
  - Uniform prior $\alpha_k = 1$ (Laplace), $\alpha_k = \frac{1}{2}$ (Jeffreys),
  - Pseudo-counts

Bayesian Estimator for a Categorical Distribution (3/3)

- The BE is given by the counts of the categories **plus some prior knowledge**.
- Pseudo counts can be used to represent our prior knowledge.

rain	no	yes	total
	5	2	7

- We can also use probabilities values directly and normalize them.

- Dataset

	rain
1	yes
2	no
3	no

- Counts

rain	no	yes	total
	$2 + 5$	$1 + 2$	$3 + 7$

- Probability - `P(rain)`

rain	no	yes	total
	70%	30%	100%

Estimating in Presence of Missing Values

- When some values are missing there are **various strategies** that can be used to deal with them.
- If we have enough values, we can **delete them**.

- Dataset

	`rain`
1	`yes`
~~2~~	~~`?`~~
3	`no`

- Counts

`rain`	`no`	`yes`	total
	1	1	2

- Probability - `P(rain)`

`rain`	`no`	`yes`	total
	50%	50%	100%

Conditional Probability Estimation

- What about **conditioning**?
- We estimate **conditional counts**.

- Dataset

	rain	wind
1	yes	yes
2	no	no
3	no	yes

- Counts

rain	no	yes	total
wind = no	1	0	1
wind = yes	1	1	2

- Probability - `P(rain | wind)`

rain	no	yes	total
wind = no	100%	0%	100%
wind = yes	50%	50%	100%

Inference

Inference: Approximate & Exact

- **Inference** in Bayesian Networks involves computing the posterior probability distribution of a set of query variables given some evidence.
- Two main types of inference:
  1. **Exact Inference**: Computes the exact posterior probabilities. Methods include variable elimination and the junction tree algorithm.
  2. **Approximate Inference**: Provides an approximation of the posterior probabilities. Methods include sampling techniques like Gibbs sampling and variational inference.

Exact Inference

- Exact inference guarantees correct results but can be computationally expensive.
- **Variable Elimination**: Systematically eliminates variables from the network to compute the marginal probability.
- **Junction Tree Algorithm**: Transforms the network into a tree structure to facilitate efficient computation.

Approximate Inference (1/4)

- Approximate inference methods are used when exact methods are infeasible.
- **Sampling Methods**: Generate samples from the distribution to approximate the posterior. Examples include Gibbs sampling and importance sampling.
- **Variational Inference**: Optimizes a family of distributions to approximate the posterior.

Approximate Inference (2/4)

- **Problem:** Compute posterior probability:

$$P(X_i \mid \mathbf{E} = \mathbf{e})$$

  when exact inference is intractable due to large or densely connected networks.
- **Idea:** Approximate $P(X_i \mid \mathbf{E})$ using sampling or variational methods.
  - **a) Monte Carlo Sampling:**
    - Draw samples from the joint or conditional distribution.
    - Estimate expectations by empirical averages:

$$\hat{P}(X_i \mid \mathbf{E} = \mathbf{e}) \approx \frac{1}{N} \sum_{j=1}^{N} \mathbb{1}\{X_i^{(j)} = x_i\}.$$

Approximate Inference (3/4)

- **Problem:** Compute posterior probability:

$$P(X_i \mid \mathbf{E} = \mathbf{e})$$

  when exact inference is intractable due to large or densely connected networks.
- **Idea:** Approximate $P(X_i \mid \mathbf{E})$ using sampling or variational methods.
  - **b) Variational Inference:**
    - Approximate posterior with a simpler family $Q(X_i)$.
    - Optimize by minimizing the KL divergence:

$$Q^*(X_i) = \arg\min_Q \mathrm{KL}\big(Q(X_i) \,\|\, P(X_i \mid \mathbf{E})\big).$$

Approximate Inference (4/4)

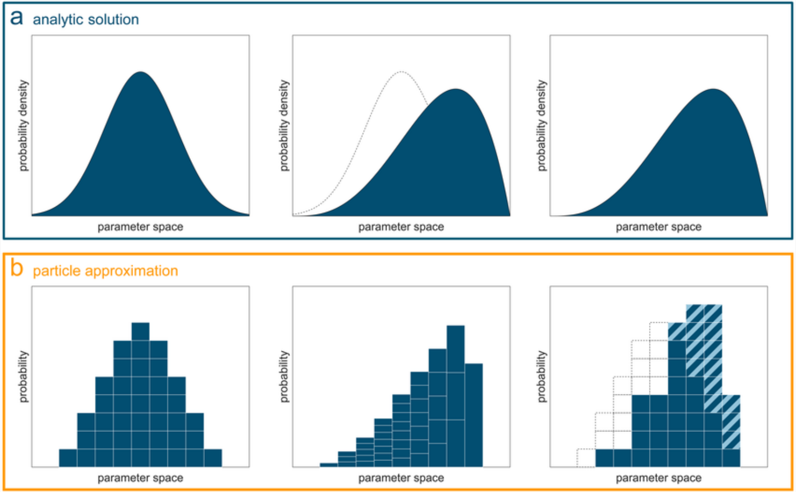


Figure 1: Example of Approximate Inference

Structure Learning

Structure Learning

- Structure learning is the task of identifying the optimal Directed Acyclic Graph (DAG) that captures the dependencies among variables.
- Two main approaches:
    - **Constraint-based:** Uses statistical tests to infer independence relationships.
    - **Score-based:** Searches for the best structure according to a scoring function.
- Hybrid methods combine both approaches to leverage their strengths.

Scoring Criteria

- Scoring functions evaluate how well a structure fits the data.
- Common scoring criteria:
  - **BIC (Bayesian Information Criterion)**: Penalizes model complexity.

$$\mathrm{BIC}(\mathcal{G} \mid \mathbf{D}) = -2\log L(\hat{\boldsymbol{\theta}}) + k\log n$$

  - **AIC (Akaike Information Criterion)**: Balances goodness of fit.

$$\mathrm{AIC}(\mathcal{G} \mid \mathbf{D}) = -2\log L(\hat{\boldsymbol{\theta}}) + 2k$$

  - **Bayesian Dirichlet (BD) scores**: Use priors and likelihood.

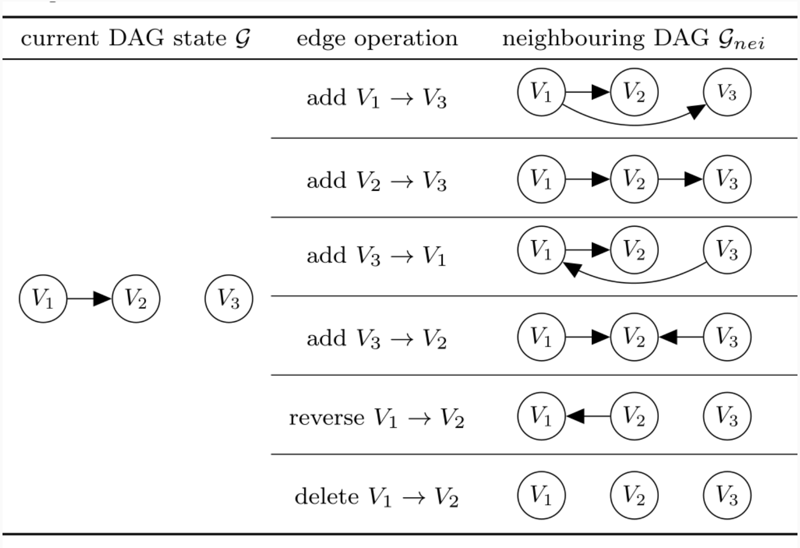
$$\mathrm{BD}(\mathcal{G} \mid \mathbf{D}) = \prod_{i=1}^{m}\prod_{j=1}^{q_i}\frac{\Gamma(\alpha_{ij})}{\Gamma(\alpha_{ij}+N_{ij})}\prod_{k=1}^{r_i}\frac{\Gamma(\alpha_{ijk}+N_{ijk})}{\Gamma(\alpha_{ijk})}$$

- The higher the score, the better the structure (depending on criterion).

Hill-Climbing Algorithm

- A greedy search algorithm used in score-based structure learning.
    1. Starts with an initial network (often empty or naive Bayes).
    2. Iteratively applies local changes (add, remove, reverse an edge).
    3. Compute the score improvement.
    4. Stops when no further improvement is possible.
- Many local optima, restarts or tabu search can mitigate the problem.

Hill-Climbing Algorithm



Expert's Knowledge - Forbidden and Required Edges

- Domain experts can specify constraints:
  - **Forbidden edges:** Edges that must not appear in the network.
  - **Required edges:** Edges that must be included.
- These constraints guide the learning process, improve interpretability, and reduce the search space.
- Especially useful in domains with strong prior knowledge (e.g., medicine, ...).

Expert's Knowledge - Temporal Order

- In many domains, the temporal order of variables is known (cause precedes effect).
- Temporal constraints can enforce a **partial ordering over nodes**.
- Prevents cycles and improves the plausibility of learned structures.
- Common in time-series, decision support systems, and diagnosis models.