Complete API Documentation

Reference Manual: Lossless Bayesian Network

Shyamal Chandra

2025

Abstract

This reference manual provides complete API documentation for the Lossless Bayesian Network implementation. It includes detailed descriptions of all classes, methods, parameters, return values, and usage examples.

Contents

1 Introduction

This reference manual documents the complete API of the Lossless Bayesian Network implementation. All classes, methods, and their usage are described in detail.

2 Header Files

2.1 node.hpp

Defines the `Node` class representing a variable in the Bayesian network.

2.2 cpt.hpp

Defines the `ConditionalProbabilityTable` class for storing conditional probability distributions.

2.3 bayesian_network.hpp

Defines the `BayesianNetwork` class, the main interface for working with Bayesian networks.

3 Node Class

3.1 Overview

The `Node` class represents a variable in the Bayesian network with its possible states and parent relationships.

3.2 Public Members

Member	Type	Description
`name`	`std::string`	Variable name identifier
`states`	`std::vector<std::string>`	Vector of possible state names
`parentIds`	`std::set<std::string>`	Set of parent node IDs
`stateIndexMap`	`std::map<std::string, int>`	Mapping state names to indices

3.3 Constructors

3.3.1 Node()

Default constructor. Creates an empty node.

Parameters: None

Returns: Node instance

3.3.2 Node(const std::string& nodeName, const std::vector<std::string>& nodeStates)

Constructor with name and states.

Parameters:

- `nodeName`: Name of the node
- `nodeStates`: Vector of possible state names

Returns: Node instance with initialized states

3.4 Methods

3.4.1 getStateIndex(const std::string& stateName) const

Get the index of a state by name.

Parameters:

- `stateName`: Name of the state

Returns: Index of the state (0-based), or -1 if not found

Example:

```
Node node("Disease", {"None", "Cold", "Flu"});
int idx = node.getStateIndex("Cold");  // Returns 1
```

3.4.2 hasState(const std::string& stateName) const

Check if a state exists.

Parameters:

- `stateName`: Name of the state

Returns: `true` if state exists, `false` otherwise

3.4.3 getNumStates() const

Get number of possible states.

Parameters: None

Returns: Number of states

3.4.4 addParent(const std::string& parentId)

Add a parent node.

Parameters:

- `parentId`: ID of the parent node

Returns: void

3.4.5 removeParent(const std::string& parentId)

Remove a parent node.

Parameters:

- `parentId`: ID of the parent node

Returns: void

3.4.6 hasParent(const std::string& parentId) const

Check if node has a specific parent.

Parameters:

- `parentId`: ID of the parent node

Returns: `true` if parent exists, `false` otherwise

3.4.7 getNumParents() const

Get number of parent nodes.
Parameters: None
Returns: Number of parents

4 ConditionalProbabilityTable Class

4.1 Overview

The `ConditionalProbabilityTable` class stores conditional probabilities in a lossless, exact representation using multi-dimensional indexing.

4.2 Constructors

4.2.1 ConditionalProbabilityTable()

Default constructor. Creates an empty CPT.
Parameters: None
Returns: CPT instance

4.2.2 ConditionalProbabilityTable(const std::vector<size_t>& dims)

Constructor with dimensions.
Parameters:

- `dims`: Vector of dimensions (last is node, others are parents)

Returns: CPT instance with initialized dimensions
Example:

```
// CPT for node with 2 parents (3 states each) and 2 node states
std::vector<size_t> dims = {3, 3, 2};
ConditionalProbabilityTable cpt(dims);
```

4.3 Methods

4.3.1 setProbability(const std::vector<size_t>& parentStates, size_t nodeState, double prob)

Set probability for given parent and node state indices.
Parameters:

- `parentStates`: Vector of parent state indices
- `nodeState`: Index of node state
- `prob`: Probability value (must be in [0, 1])

Returns: void
Throws: `std::runtime_error` if probability is out of range or indices are invalid
Example:

```
// P(node=1 | parent1=0, parent2=2) = 0.75
cpt.setProbability({0, 2}, 1, 0.75);
```

4.3.2 getProbability(const std::vector<size_t>& parentStates, size_t nodeState) const

Get probability for given parent and node state indices.

Parameters:

- `parentStates`: Vector of parent state indices
- `nodeState`: Index of node state

Returns: Probability value
Throws: `std::runtime_error` if indices are invalid

4.3.3 normalize()

Normalize probabilities for each parent configuration. Ensures each conditional distribution sums to 1.0.

Parameters: None
Returns: void
Example:

```
cpt.setProbability({0}, 0, 0.6);
cpt.setProbability({0}, 1, 0.4);
cpt.normalize();  // Ensures sum = 1.0
```

4.3.4 isValid(double tolerance = 1e-6) const

Validate that all conditional distributions sum to 1.0.

Parameters:

- `tolerance`: Tolerance for floating point comparison (default: 1e-6)

Returns: `true` if valid, `false` otherwise

4.3.5 getDimensions() const

Get dimensions of the CPT.

Parameters: None
Returns: Vector of dimensions

4.3.6 getTotalSize() const

Get total number of probability entries.

Parameters: None
Returns: Total size

5 BayesianNetwork Class

5.1 Overview

The `BayesianNetwork` class is the main interface for working with Bayesian networks. It provides network construction, inference, and I/O capabilities.

5.2 Constructors

5.2.1 BayesianNetwork()

Default constructor. Creates an empty network.

Parameters: None

Returns: BayesianNetwork instance

5.3 Network Construction Methods

5.3.1 addNode(const std::string& nodeId, const std::string& nodeName, const std::vector<std::string>& states)

Add a node to the network.

Parameters:

- `nodeId`: Unique identifier for the node
- `nodeName`: Name of the node
- `states`: Vector of possible state names

Returns: void

Throws: `std::runtime_error` if node ID already exists

Example:

```
network.addNode("Disease", "Disease", {"None", "Cold", "Flu"});
```

5.3.2 addEdge(const std::string& parentId, const std::string& childId)

Add an edge from parent to child.

Parameters:

- `parentId`: ID of parent node
- `childId`: ID of child node

Returns: void

Throws: `std::runtime_error` if:

- Parent or child node does not exist
- Adding edge would create a cycle
- Attempting to add self-loop

Example:

```
network.addEdge("Disease", "Symptom");
```

5.3.3 setCPT(const std::string& nodeId, const ConditionalProbabilityTable& cpt)

Set conditional probability table for a node.

Parameters:

- `nodeId`: ID of the node
- `cpt`: Conditional probability table

Returns: void

Throws: `std::runtime_error` if node does not exist

5.4 Inference Methods

5.4.1 getConditionalProbability(const std::string& nodeId, const std::string& nodeState, const std::map<std::string, std::string>& parentStates) const

Get conditional probability.

Parameters:

- `nodeId`: ID of the node
- `nodeState`: State of the node
- `parentStates`: Map of parent IDs to their states

Returns: Conditional probability P(nodeState | parentStates)
Throws: `std::runtime_error` if:

- Node does not exist
- CPT not set for node
- Missing or invalid parent states
- Invalid node state

5.4.2 computeJointProbability(const std::map<std::string, std::string>& assignment) const

Compute joint probability for a full assignment.

Parameters:

- `assignment`: Map of node IDs to their states

Returns: Joint probability P(assignment)
Throws: `std::runtime_error` if assignment is incomplete
Example:

```
std::map<std::string, std::string> assignment;
assignment["Disease"] = "Flu";
assignment["Symptom"] = "Yes";
double prob = network.computeJointProbability(assignment);
```

5.4.3 variableElimination(const std::vector<std::string>& queryNodes, const std::map<std::stri std::string>& evidence) const

Variable elimination for exact inference.

Parameters:

- `queryNodes`: Vector of node IDs to query
- `evidence`: Map of observed node IDs to their states

Returns: Map from query assignments to their probabilities (normalized)
Example:

```
std::map<std::string, std::string> evidence;
evidence["Symptom"] = "Yes";
std::vector<std::string> query = {"Disease"};
auto results = network.variableElimination(query, evidence);
```

5.5 Utility Methods

5.5.1 getNode(const std::string& nodeId) const

Get node by ID.

Parameters:

- `nodeId`: ID of the node

Returns: Const reference to the node
Throws: `std::runtime_error` if node does not exist

5.5.2 getNodeIds() const

Get all node IDs.

Parameters: None
Returns: Vector of node IDs

5.5.3 generateAssignments(const std::vector<std::string>& nodeIds, std::vector<std::map<std:: std::string>& assignments) const

Generate all possible assignments for given nodes.

Parameters:

- `nodeIds`: Vector of node IDs
- `assignments`: Output vector of assignments

Returns: void
Throws: `std::runtime_error` if any node does not exist

5.6 File I/O Methods

5.6.1 saveToFile(const std::string& filename) const

Save network to file.

Parameters:

- `filename`: Output filename

Returns: void
Throws: `std::runtime_error` if file cannot be opened

5.6.2 loadFromFile(const std::string& filename)

Load network from file.

Parameters:

- `filename`: Input filename

Returns: void
Throws: `std::runtime_error` if file cannot be loaded

6 Error Handling

All methods that can fail throw `std::runtime_error` exceptions with descriptive error messages. Common error conditions include:

- Node does not exist
- Invalid state names
- Cycle detection (invalid DAG)
- Invalid probability values
- Missing CPTs
- File I/O errors

7 Usage Examples

7.1 Complete Example

```cpp
#include "bayesian_network.hpp"
#include <iostream>

int main() {
    BayesianNetwork network;

    // Add nodes
    network.addNode("Disease", "Disease", {"None", "Cold", "Flu"});
    network.addNode("Symptom", "Fever", {"No", "Yes"});

    // Add edge
    network.addEdge("Disease", "Symptom");

    // Create CPT
    std::vector<size_t> dims = {3, 2};
    ConditionalProbabilityTable cpt(dims);
    cpt.setProbability({0}, 0, 0.9);  // P(No|None)
    cpt.setProbability({0}, 1, 0.1);  // P(Yes|None)
    cpt.setProbability({1}, 0, 0.7);  // P(No|Cold)
    cpt.setProbability({1}, 1, 0.3);  // P(Yes|Cold)
    cpt.setProbability({2}, 0, 0.2);  // P(No|Flu)
    cpt.setProbability({2}, 1, 0.8);   // P(Yes|Flu)
    cpt.normalize();
    network.setCPT("Symptom", cpt);

    // Prior for Disease
    std::vector<size_t> diseaseDims = {3};
    ConditionalProbabilityTable diseaseCPT(diseaseDims);
    diseaseCPT.setProbability({}, 0, 0.7);
    diseaseCPT.setProbability({}, 1, 0.2);
    diseaseCPT.setProbability({}, 2, 0.1);
    diseaseCPT.normalize();
    network.setCPT("Disease", diseaseCPT);

    // Inference
    std::map<std::string, std::string> evidence;
    evidence["Symptom"] = "Yes";
```

```
    std::vector<std::string> query = {"Disease"};
    auto results = network.variableElimination(query, evidence);

    // Display results
    for (const auto& pair : results) {
        std::cout << "P(Disease=" << pair.first.at("Disease")
                  << ") = " << pair.second << std::endl;
    }

    return 0;
}
```

8 Performance Notes

- Variable elimination has exponential time complexity in the worst case
- CPT storage is exponential in the number of parents
- Topological ordering minimizes computation during inference
- State lookup is O(1) via hash maps

9 Copyright