Quantum Bayesian Networks

version

Oleksandr Hoviadin, Mohamed Rayan Amara, Siwar Sraieb

June 24, 2023

Contents

Welcome to Quantum Bayesian Networks's documentation! 1

QuantumBayesian 1

QuantumBayesian package 1

Submodules 1

QuantumBayesian.inference_exact module 1

QuantumBayesian.inference_jt module 2

QuantumBayesian.inference module 3

QuantumBayesian.calcule module 3

QuantumBayesian.qbn module 4

Module contents 5

Indices and tables 5

Index 7

Python Module Index 9

Welcome to Quantum Bayesian Networks's documentation!

QuantumBayesian

QuantumBayesian package

Submodules

QuantumBayesian.inference_exact module

class QuantumBayesian.inference_exact.**Inference_Exact** (qbn)
Bases: **Inference**
Class Inference_Exact represents exact inference in Quantum Bayesian Network.
Inference_Exact(qbn) -> Inference_Exact

Parameters: **qbn** (*QBN*) – the Quantum Bayesian Network in which we want to compute inference.

addEvidence (var, val)
Adds evidence to the Quantum Bayesian Network by specifying the observed value for a variable.

Parameters:
- **var** (*str*) – The name of the variable for which evidence is provided.
- **val** – The observed value for the variable.

addTarget (var)
Adds a target variable to the list of targets.

Parameters: **var** (*str*) – The name of the target variable to add.

calcul (vals)
Calculates the complex sum of probabilities for the given variable values.

Parameters: **vals** (*dict*) – A dictionary containing the values of variables.
Returns: A tuple containing the module and argument of the complex sum.
Return type: tuple (float, float)

currentEvidence ()
Get the current evidence in the Quantum Bayesian Network.

Returns: A dictionary containing the evidence variables and their observed values.
Return type: dict

currentTargets ()
Get the current list of target variables.

Returns: The list of target variables.
Return type: list [str]

makeInference ()
Performs the inference in the Quantum Bayesian Network based on the given evidence.

posterior (var)
Computes the posterior probability distribution of a variable.

Parameters: **var** (*str*) – The name of the variable.

Returns: The posterior probability distribution.
Return type: gum.Potential

posteriorJoint (`variables`**)**
Computes the joint posterior probability distribution of a set of variables.

Parameters: **variables** (*list[str]*) – The names of the variables.
Returns: The joint posterior probability distribution.
Return type: gum.Potential

removeEvidence (`var`**)**
Removes the evidence for a specific variable from the Quantum Bayesian Network.

Parameters: **var** (*str*) – The name of the variable for which to remove the evidence.

removeTarget (`var`**)**
Removes a target variable from the list of targets.

Parameters: **var** (*str*) – The name of the target variable to remove.

QuantumBayesian.inference_jt module

class `QuantumBayesian.inference_jt.`**Inference_JT** (qbn)
Bases: **Inference**
Class Inference_JT represents inference in Quantum Bayesian Network computed using optimized message passing algorithm.
Inference_JT(qbn) -> Inference_JT

Parameters: **qbn** (*QBN*) – the Quantum Bayesian Network in which we want to compute inference.

addEvidence (`var`**,** `val`**)**
Adds evidence to the Quantum Bayesian Network by specifying the observed value for a variable.

Parameters:
- **var** (*str*) – The name of the variable for which evidence is provided.
- **val** – The observed value for the variable.

calculMessage (`clique_send`**,** `clique_receive`**)**
Calculates the message to be sent from a sending clique to a receiving clique.

Parameters:
- **clique_send** (*int*) – Identifier of the sending clique.
- **clique_receive** (*int*) – Identifier of the receiving clique.

Returns: Tuple containing the message potentials for the model part and argument part.
Return type: tuple (gum.Potential, gum.Potential)

currentEvidence ()
Get the current evidence in the Quantum Bayesian Network.

Returns: A dictionary containing the evidence variables and their observed values.
Return type: dict

makeInference ()
Performs inference using the optimized message passing algorithm.

makeJt ()
Constructs the junction tree for the Quantum Bayesian Network.

posterior (`var`**)**

Computes the posterior probability distribution of a variable.

Parameters: **var** (*str*) – Name of the variable.
Returns: The posterior probability distribution.
Return type: gum.Potential

removeEvidence (var**)**
Removes the evidence for a specific variable from the Quantum Bayesian Network.

Parameters: **var** (*str*) – The name of the variable for which to remove the evidence.

sendMessage (clique_send**,** clique_receive**)**
Sends a message from a sending clique to a receiving clique in the junction tree.

Parameters:
- **clique_send** (*int*) – Identifier of the sending clique.
- **clique_receive** (*int*) – Identifier of the receiving clique.

setPotentials ()
Sets the potentials for the nodes in the junction tree.

showJt ()
Displays the junction tree of the Quantum Bayesian Network.

QuantumBayesian.inference module

class QuantumBayesian.inference.**Inference**
Bases: **object**

addEvidence ()

currentEvidence ()

makeInference ()

posterior ()

removeEvidence ()

QuantumBayesian.calcule module

QuantumBayesian.calcule.**complex_to_polar** (z)
Converts a complex number to polar coordinates.

Parameters: **z** (*complex*) – The complex number.
Returns: A tuple containing the module and argument of the complex number in radians.
Return type: tuple (float, float)

QuantumBayesian.calcule.**create_complex_number** (module, argument)
Creates a complex number given its module and argument.

Parameters:
- **module** (*float*) – The module of the complex number.
- **argument** (*float*) – The argument of the complex number in radians.

Returns: The complex number.
Return type: complex

QuantumBayesian.calcule.**exp_to_theta** (bn, cpt)
Convert the arguments (theta) in a Conditional Probability Table (CPT) from exponentiated form to log-odds.

Parameters:
- **bn** (*gum.BayesNet*) – The Bayesian network containing the CPT.
- **cpt** (*gum.Potential*) – The Conditional Probability Table to convert.

`QuantumBayesian.calcule.`**`normalize_cpt`**`(bn, cpt)`
Normalize the probabilities in a Conditional Probability Table (CPT) so that they sum up to 1.

Parameters:
- **bn** (*gum.BayesNet*) – The Bayesian network containing the CPT.
- **cpt** (*gum.Potential*) – The Conditional Probability Table to normalize.

`QuantumBayesian.calcule.`**`normalize_list`**`(lst)`
Normalizes a list of numbers by dividing each element by the sum of all elements.

Parameters: **lst** (*list*) – The list of numbers.
Returns: The normalized list.
Return type: list

`QuantumBayesian.calcule.`**`produit_complexes`**`(module1, argument1, module2, argument2)`
Computes the product of two complex numbers given their modules and arguments.

Parameters:
- **module1** (*float*) – The module of the first complex number.
- **argument1** (*float*) – The argument of the first complex number in radians.
- **module2** (*float*) – The module of the second complex number.
- **argument2** (*float*) – The argument of the second complex number in radians.

Returns: The product of the two complex numbers.
Return type: complex

`QuantumBayesian.calcule.`**`theta_to_exp`**`(bn, cpt)`
Convert the arguments (theta) in a Conditional Probability Table (CPT) to exponentiated form.

Parameters:
- **bn** (*gum.BayesNet*) – The Bayesian network containing the CPT.
- **cpt** (*gum.Potential*) – The Conditional Probability Table to convert.

QuantumBayesian.qbn module

class `QuantumBayesian.qbn.`**`QBN`**
Bases: **`object`**
Class QBN represents a Quantum Bayesian Network.

`add` **(**`varname`**,** `nbrmod`**)**
Adds a new variable to the network.

Parameters:
- **varname** (*str*) – The name of the variable to be added.
- **nbrmod** (*int*) – The number of modules associated with the variable.

`addArc` **(**`var1`**,** `var2`**)**
Adds a new arc between the given variables to the network.

Parameters:
- **var1** (*str*) – The name of the first variable.
- **var2** (*str*) – The name of the second variable.

`argument` **(**`var`**)**
Returns the conditional probability table associated with the argument of the given variable.

Parameters: **var** (*str*) – The name of the variable.
Returns: The conditional probability table (CPT) associated with the argument of the variable.

Return type: gum.Potential

listNodes ()
Returns the list of nodes in the network.

Returns: The list of node names in the network.
Return type: list [str]

module (var)
Returns the conditional probability table associated with the module of the given variable.

Parameters: **var** (*str*) – The name of the variable.
Returns: The conditional probability table (CPT) associated with the module of the variable.
Return type: gum.Potential

showQBN ()
Displays the network.

Returns: The Bayesian network object representing the network.
Return type: gum.BayesNet

verifcpt (var)
Verifies the validity of the conditional probability table (CPT) of the given variable: 1) Each module value in each row of the table should be positive and not exceed 1. 2) The sum of their squares should be equal to 1.

Parameters: **var** (*str*) – The name of the variable.
Returns: True if the CPT is valid, False otherwise.
Return type: bool

Module contents

Indices and tables

- **genindex**
- **modindex**
- **search**

Index

A

add() (QuantumBayesian.qbn.QBN method)

addArc() (QuantumBayesian.qbn.QBN method)

addEvidence() (QuantumBayesian.inference.Inference method)

(QuantumBayesian.inference_exact.Inference_Exact method)

(QuantumBayesian.inference_jt.Inference_JT method)

addTarget() (QuantumBayesian.inference_exact.Inference_Exact method)

argument() (QuantumBayesian.qbn.QBN method)

C

calcul() (QuantumBayesian.inference_exact.Inference_Exact method)

calculMessage() (QuantumBayesian.inference_jt.Inference_JT method)

complex_to_polar() (in module QuantumBayesian.calcule)

create_complex_number() (in module QuantumBayesian.calcule)

currentEvidence() (QuantumBayesian.inference.Inference method)

(QuantumBayesian.inference_exact.Inference_Exact method)

(QuantumBayesian.inference_jt.Inference_JT method)

currentTargets() (QuantumBayesian.inference_exact.Inference_Exact method)

E

exp_to_theta() (in module QuantumBayesian.calcule)

I

Inference (class in QuantumBayesian.inference)

Inference_Exact (class in QuantumBayesian.inference_exact)

Inference_JT (class in QuantumBayesian.inference_jt)

L

listNodes() (QuantumBayesian.qbn.QBN method)

M

makeInference() (QuantumBayesian.inference.Inference method)

(QuantumBayesian.inference_exact.Inference_Exact method)

(QuantumBayesian.inference_jt.Inference_JT method)

makeJt() (QuantumBayesian.inference_jt.Inference_JT method)

module

QuantumBayesian

QuantumBayesian.calcule

QuantumBayesian.inference

QuantumBayesian.inference_exact

QuantumBayesian.inference_jt

QuantumBayesian.qbn

module() (QuantumBayesian.qbn.QBN method)

N

normalize_cpt() (in module QuantumBayesian.calcule)

normalize_list() (in module QuantumBayesian.calcule)

P

posterior() (QuantumBayesian.inference.Inference method)

(QuantumBayesian.inference_exact.Inference_Exact method)

(QuantumBayesian.inference_jt.Inference_JT method)

posteriorJoint() (QuantumBayesian.inference_exact.Inference_Exact method)

produit_complexes() (in module QuantumBayesian.calcule)

Q

QBN (class in QuantumBayesian.qbn)

QuantumBayesian

module

QuantumBayesian.calcule

module

QuantumBayesian.inference

module

QuantumBayesian.inference_exact

module

QuantumBayesian.inference_jt

module

QuantumBayesian.qbn

module

R

removeEvidence() (QuantumBayesian.inference.Inference method)

(QuantumBayesian.inference_exact.Inference_Exact method)

(QuantumBayesian.inference_jt.Inference_JT method)

removeTarget() (QuantumBayesian.inference_exact.Inference_Exact method)

S

sendMessage() (QuantumBayesian.inference_jt.Inference_JT method)

setPotentials() (QuantumBayesian.inference_jt.Inference_JT method)

showJt() (QuantumBayesian.inference_jt.Inference_JT method)

showQBN() (QuantumBayesian.qbn.QBN method)

T

theta_to_exp() (in module QuantumBayesian.calcule)

V

verifcpt() (QuantumBayesian.qbn.QBN method)

Python Module Index

q

QuantumBayesian

QuantumBayesian.calcule

QuantumBayesian.inference

QuantumBayesian.inference_exact

QuantumBayesian.inference_jt

QuantumBayesian.qbn