PySCMs

Release 2.0.0

Black-Swan-ICL

Apr 15, 2021

CONTENTS:

CHAPTER

ONE

STRUCTURALCAUSALMODELS

1.1 StructuralCausalModels package

1.1.1 Submodules

1.1.2 StructuralCausalModels.dag module

exception StructuralCausalModels.dag.**TopologicalOrderingMethodNotImplemented**
Bases: Exception

Raised when the requested topological ordering method is not implemented.

exception StructuralCausalModels.dag.**InvalidOrdering**
Bases: Exception

Raised when an ordering is not a valid causal ordering for a given DAG.

class StructuralCausalModels.dag.**DirectedAcyclicGraph**(*adjacency_matrix*, *name=''*)
Bases: *StructuralCausalModels.directed_graph.DirectedGraph*

A class to represent Directed Acyclic Graphs (DAGs).

A DirectedAcyclicGraph object is a representation of a DAG. It is defined by the adjacency matrix of the graph ; validation will be performed to check that the graph defined is directed and acyclic. An exception will be raised otherwise.

Parameters

- **adjacency_matrix** (*array_like*) – The adjacency matrix of the graph.
- **name** (*str, optional*) – The name of the object created (default is ‘’).

Raises ***InvalidAdjacencyMatrix*** – If the adjacency matrix does not define a directed and acyclic graph.

static validate_dag_adjacency_matrix(*matrix*, *atol=1e-06*)
Checks that a matrix is a valid adjacency matrix for a directed acyclic graph. Uses the characterisation of acyclicity established by D. Wei, T. Gao and Y. Yu in[1] : a directed graph is acyclic if and only if its adjacency matrix is nilpotent.

Parameters

- **matrix** (*array_like*) – The matrix to check.

[1] Wei, D., Gao, T. and Yu, Y. “DAGs with No Fears : A Closer Look at Continuous Optimization for Learning Bayesian Networks”. *Advances in Neural Information Processing Systems*, volume 33, pp. 3895-3906, 2020.

- **atol** (*float, optional*) – The absolute tolerance used to check that the eigenvalues are all equal to 0 (default is 10^{-6}).

Returns *bool* – Whether the matrix to check is a valid adjacency matrix for a directed acyclic graph.

Notes

kahn_algorithm()
An implementation of Kahn's algorithm for topological ordering of a graph.

Returns *list* – A topological ordering of the graph.

compute_causal_order(*method='kahn'*)
Computes a causal order of the DAG using the method chosen by the user.

Parameters **method** (*str, optional*) – The method to use to compute the causal order (default is 'kahn').

Returns *list* – A causal order of the DAG.

Raises ***TopologicalOrderingMethodNotImplemented*** – If the method chosen by the usr is not implemented.

check_topological_ordering(*ordering*)
Checks whether an ordering is a correct topological ordering for the graph.

Parameters **ordering** (*list*) – The candidate ordering.

Returns *bool* – Whether the ordering is a correct topological ordering for the graph.

Raises ***InvalidOrdering*** – If the ordering passed is not a valid ordering for the graph.

static causal_order_to_dag(*causal_order*)
Generates the maximally connected DAG that is compatible with the causal order provided i.e. for all j such that $j > i$ in the causal order, there will be an edge from X_i to X_j in the DAG (i.e. there will be a 1 in position $[i, j]$ in the DAG's adjacency matrix).

Parameters **causal_order** (*array_like*) – The causal order.

Returns *DirectedAcyclicGraph* – The maximally connected DAG compatible with the causal order.

1.1.3 StructuralCausalModels.directed_graph module

class StructuralCausalModels.directed_graph.**DirectedGraph**(*adjacency_matrix*, *name=''*)
Bases: *StructuralCausalModels.graph.Graph*

A class to represent directed graphs.

A DirectedGraph object is a representation of a directed graph. It is defined by the adjacency matrix of the graph ; validation will be performed to check that the graph defined is directed. An exception will be raised otherwise.

Parameters

- **adjacency_matrix** (*array_like*) – The adjacency matrix of the graph.
- **name** (*str, optional*) – The name of the object created (default is '').

Raises ***InvalidAdjacencyMatrix*** – If the adjacency matrix does not define a directed graph.

static validate_directed_graph_adjacency_matrix(*matrix*)
Checks that a matrix is a valid adjacency matrix for a directed graph.

Parameters **matrix** (*array_like*) – The matrix to check.

Returns *bool* – Whether the matrix to check is a valid adjacency matrix for a directed graph.

1.1.4 StructuralCausalModels.graph module

class StructuralCausalModels.graph.**Graph**(*adjacency_matrix*, *name=''*)
Bases: object

A class to represent graphs.

A Graph object is a representation of the graph. The graph may be very general : it does not have to be directed, or acyclic for instance. It is arguably easiest to think of a graph in terms of its adjacency matrix which is why this implementation is 'adjacency matrix'-driven. Other representations can be more convenient depending on context, which is why these representations are automatically generated upon construction and stored as attributes of the Graph object.

Parameters

- **adjacency_matrix** (*array_like*) – The adjacency matrix of the graph.
- **name** (*str, optional*) – The name of the object created (default is '').

adjacency_matrix_representation
The representation of the graph based on the adjacency matrix.

Type *GraphViaAdjacencyMatrix*

adjacency_list_representation
The representation of the graph based on the adjacency lists.

Type *GraphViaAdjacencyLists*

edge_representation
The adjacency of the graph based on the typed (undirected, forward etc.) edges.

Type *GraphViaEdges*

static validate_binary_matrix(*matrix*)
Validates that a matrix is a proper adjacency matrix.

A proper adjacency matrix contains only 0's and 1's.

Parameters **matrix** (*array_like*) – The matrix to validate.

Returns *bool* – Whether the matrix is a valid adjacency matrix.

property name
the name of the graph.

Type str

property adjacency_matrix
the adjacency matrix of the graph.

Type array_like

structural_hamming_distance(*other*, *penalty_edge_mismatch_func=None*)
Computes the Structural Hamming Distance between two graphs.

By default, the Structural Hamming Distance (SHD) is equal to the number of edges in the graphs that are not of the same type. A different weighted scheme for penalty computation may be provided (we may

want to penalise the presence of an edge in the opposite direction more than the absence of an edge, for example).

Parameters

- **other** (*Graph*) – The graph to compare.
- **penalty_edge_mismatch_func** (*callable, optional*) – The edge mismatch penalty scheme (default is None in which case a built-in function is used).

Returns *float* – The Structural Hamming Distance (SHD), in its default form or under a user-provided edge mismatch penalty scheme.

Raises ***GraphsCannotBeCompared*** – Raised if the graphs do not have the same vertex set.

static adjacency_matrix_to_adjacency_lists (*adjacency_matrix*)
Converts an adjacency matrix to the corresponding adjacency lists.

Parameters **adjacency_matrix** (*array_like*) – The adjacency matrix.

Returns *list* – The adjacency lists.

static adjacency_lists_to_adjacency_matrix (*adjacency_lists*)
Converts adjacency lists to the corresponding adjacency matrix.

Parameters **adjacency_lists** (*list*) – The adjacency lists.

Returns *array_like* – The adjacency matrix.

static compute_edge_type (*m_ij*, *m_ji*)
Determines the type of the edge between two vertices.

Determines the type of the edge between X_i and X_j based on the values of $M_{i,j}$ and $M_{j,i}$ where M is the adjacency matrix of the graph.

Parameters

- **m_ij** (*int*) – The element in the adjacency matrix at the intersection of the i-th row and the j-th column.
- **m_ji** (*int*) – The element in the adjacency matrix at the intersection of the j-th row and the i-th column.

Returns *EdgeType* – The typed edge between X_i and X_j in the graph.

static adjacency_matrix_to_edges (*adjacency_matrix*)
Converts adjacency matrix to the corresponding typed edges.

Parameters **adjacency_matrix** (*array_like*) – The adjacency matrix.

Returns

dict – The typed edges.

The keys are tuples (i, j) ; they indicates the edge is between X_i and X_j in the graph. The values are EdgeType objects which indicate what type of edge is between X_i and X_j in the graph.

static edges_to_adjacency_lists (*edges*)
Converts the typed edges to the corresponding adjacency lists.

Parameters **edges** (*dict*) – The typed edges.

The keys are tuples (i, j) ; they indicates the edge is between X_i and X_j in the graph. The values are EdgeType objects which indicate what type of edge is between X_i and X_j in the graph.

Returns *list* – The adjacency lists.

static edges_to_adjacency_matrix(*edges*)
Converts the typed edges to the corresponding adjacency matrix.

Parameters **edges** (*dict*) – The typed edges.

The keys are tuples (i, j) ; they indicates the edge is between X_i and X_j in the graph. The values are EdgeType objects which indicate what type of edge is between X_i and X_j in the graph.

Returns *array_like* – The adjacency matrix.

static adjacency_lists_to_edges(*adjacency_lists*)
Converts adjacency lists to the corresponding typed edges..

Parameters **adjacency_lists** (*list*) – The adjacency lists.

Returns

dict – The typed edges.

The keys are tuples (i, j) ; they indicates the edge is between X_i and X_j in the graph. The values are EdgeType objects which indicate what type of edge is between X_i and X_j in the graph.

1.1.5 StructuralCausalModels.graph_via_adjacency_lists module

exception StructuralCausalModels.graph_via_adjacency_lists.**InvalidAdjacencyLists**
Bases: Exception

Raised when the adjacency lists are not valid for the graph.

class StructuralCausalModels.graph_via_adjacency_lists.**GraphViaAdjacencyLists**(*nb_vertices*, *adjacency_lists*, *name=''*)
Bases: object

Implements a graph structure using an adjacency list representation.

As an "extra security", the number of vertices must be provided as well as the adjacency lists ; there must be as many vertices as there are adjacency lists.

Parameters

- **nb_vertices** (*int*) – The number of vertices in the graph.
- **adjacency_lists** (*list*) – The list of adjacency lists defining the graph. The ordering in the list is the natural one : adjacency_lists[0] is the adjacency list for vertex X_0 etc.
- **name** (*str, optional*) – The name of the object created (default is '').

indegrees
The list of in-degrees of the vertices i.e. the number of edges pointing to the vertices. The ordering in the list is the natural one : indegrees[0] is the in-degree for vertex X_0 etc.

Type list

Raises ***InvalidAdjacencyLists*** – If the number of adjacency lists provided does not match the number of vertices in the graph.

1.1.6 StructuralCausalModels.graph_via_adjacency_matrix module

exception StructuralCausalModels.graph_via_adjacency_matrix.**InvalidAdjacencyMatrix**
Bases: Exception

Raised if the adjacency matrix does not contain only 0's and 1's.

class StructuralCausalModels.graph_via_adjacency_matrix.**GraphViaAdjacencyMatrix**(*adjacency_mat*, *name=''*)
Bases: object

Implements a graph structure using an adjacency matrix representation.

Parameters

- **adjacency_matrix** (*array_like*) – The adjacency matrix of the graph.
- **name** (*str, optional*) – The name of the object created (default is '').

Raises ***InvalidAdjacencyMatrix*** – If the adjacency matrix provided does not contain only 0's and 1's.

static validate_binary_matrix(*matrix*)
Checks that a matrix is an adjacency matrix (i.e. a square matrix containing only 0's and 1's).

Parameters **matrix** (*array_like*) – The matrix to check.

Returns *bool* – Whether the matrix is a valid adjacency matrix.

1.1.7 StructuralCausalModels.graph_via_edges module

class StructuralCausalModels.graph_via_edges.**EdgeType**(*value*)
Bases: enum.Enum

An enumeration to represent possible types of edges between vertices.

NONE = 'no edge'

FORWARD = '->'

BACKWARD = '<-'

UNDIRECTED = '--'

exception StructuralCausalModels.graph_via_edges.**GraphsCannotBeCompared**
Bases: Exception

Raised when two graphs do not have the same vertex set.

exception StructuralCausalModels.graph_via_edges.**ImpossibleEdgeConfiguration**
Bases: Exception

Raised when an edge is not one of the possible types of edges.

class StructuralCausalModels.graph_via_edges.**GraphViaEdges**(*edges*, *name=''*)
Bases: object

Implements a graph structure using a representation via typed edges.

Typed edges make it possible than X_i and X_j have no edge between them, or that they have an undirected edge, or a forward edge $X_i \longrightarrow X_j$ or a backward edge $X_i \longleftarrow X_j$.

Parameters

- **edges** (*dict*) – A dictionary defining the edges of the graph in the format (i, j) : EdgeType.FORWARD for $X_i \longrightarrow X_j$, for example.
- **name** (*str, optional*) – The name of the object created (default is ‘’).

static compute_penalty(*edge_1*, *edge_2*)

Provides a default method to compute the penalty incurred when two edges are of a same type or of different types.

Parameters

- **edge_1** (*EdgeType*) – The first edge.
- **edge_2** (*EdgeType*) – The second edge.

Returns *float* – The penalty incurred.

Raises ***ImpossibleEdgeConfiguration*** – If one of the edges is not one of the possible types of edges.

structural_hamming_distance(*other*, *penalty_edge_mismatch_func=None*)

Computes the Structural Hamming Distance between two graphs. By default it is equal to the number of edges in the graphs that are not of the same type. A different weighted scheme for penalty computation may be provided (we may want to penalise the presence of an edge in the opposite direction more than the absence of an edge, for example).

Parameters

- **other** (*GraphViaEdges*) – The graph to compare.
- **penalty_edge_mismatch_func** (*callable*) – The edge mismatch penalty scheme.

Returns *float* – The Structural Hamming Distance (SHD), in its default form or under a user-provided edge mismatch penalty scheme.

Raises ***GraphsCannotBeCompared*** – Raised if the graphs do not have the same vertex set.

1.1.8 StructuralCausalModels.linear_structural_causal_model module

exception StructuralCausalModels.linear_structural_causal_model.**InvalidWeightedAdjacencyMat**

Bases: Exception

Raised if the weighted adjacency matrix does not define a directed graph.

exception StructuralCausalModels.linear_structural_causal_model.**InvalidNumberOfExogenousVa**

Bases: Exception

Raised if the number of exogenous variables provided is incorrect.

By incorrect is meant that the number of exogenous variables provided is inconsistent with the number of variables expected in the SCM.

class StructuralCausalModels.linear_structural_causal_model.**LinearStructuralCausalModel**(*nam*
*nb_*
stru
tura

Bases: *StructuralCausalModels.structural_causal_model.StructuralCausalModel*

A class to represent linear Structural Causal Models.

Beware, does not check the equation are linear.

Parameters

- **nb_var** (*int*) – The number of variables in the SCM.
- **structural_equations** (*list*) – The list of the structural equations defining the SCM.
- **name** (*str, optional*) – The name of the SCM (default is ‘’).

static create_from_coefficient_matrix(*matrix*, *causal_order*, *exogenous_variables*, *name=''*)

Creates a linear SCM from a coefficient matrix.

The coefficient matrix is interpreted as the weighted adjacency matrix of the graph associated to a linear SCM.

Parameters

- **matrix** (*numpy.ndarray*) – The weighted adjacency matrix of the graph associated to the linear SCM to build.
- **causal_order** (*array_like*) – A causal order of the graph associated to the SCM.
- **exogenous_variables** (*list*) – The list of the exogenous variables. Note that the list must be ordered in the “natural” way - not necessarily in the causal ordering. In other terms, exogenous_varoables[0] contains the exogenous variables for the structural equation that has X_0 on its left-hand side etc.
- **name** (*str, optional*) – The name of the linear SCM created (default is ‘’).

Returns *LinearStructuralCausalModel* – The linear SCM corresponding to the weighted adjacency matrix provided.

Raises

- ***InvalidWeightedAdjacencyMatrix*** – If the weighted adjacency matrix provided defines a graph that is not a directed graph.
- ***InvalidNumberOfExogenousVariables*** – If the number of exogenous variables provided does not correspond to the number of variables expected from the matrix provided.

1.1.9 StructuralCausalModels.structural_causal_model module

exception StructuralCausalModels.structural_causal_model.**InconsistentStructuralCausalModel**

Bases: Exception

Raised if the Structural Causal Model is not defined in a consistent way.

exception StructuralCausalModels.structural_causal_model.**CyclicityWarning**

Bases: UserWarning

Raised if there is a high probability that the SCM contains a cycle.

exception StructuralCausalModels.structural_causal_model.**InvalidIntervention**

Bases: Exception

Raised if the intervention attempted is invalid.

class StructuralCausalModels.structural_causal_model.**StructuralCausalModel**(*nb_var*, *structural_equations*, *name=''*)

Bases: object

A class to represent Structural Causal Models (SCMs).

As an "extra security", the number of variables must be provided in addition to the structural equations ; the number of variables must be equal to the number of structural equations.

Parameters

- **nb_var** (*int*) – The number of variables in the SCM.
- **structural_equations** (*list*) – The list of the structural equations defining the SCM.
- **name** (*str, optional*) – The name of the SCM (default is '').

Raises

- ***InconsistentStructuralCausalModelDefinition*** – If the number of structural equations provided does not correspond to the number of variables in the SCM.
- ***CyclicityWarning*** – If the SCM defined is (highly likely to be) cyclic.

generate_data(*nb_samples*)

Generates samples from an SCM.

Parameters **nb_samples** (*int*) – The number of samples to generate.

Returns

pandas.DataFrame – A dataframe containing the samples.

The column names correspond to the variables in the SCM. Thus column 0 contains the samples for X_0, column 1 the samples for X_1 etc.

adjacency_matrix()

Generates the adjacency matrix of the graph corresponding to the SCM.

Returns *numpy.ndarray* – The adjacency matrix of the graph corresponding to the SCM.

compute_causal_order()

Computes a causal order of the DAG associated to the SCM.

Returns *list* – A causal order of the DAG associated to the SCM.

order_structural_equations()

Returns structural equations, ordered to follow a causal order.

The structural equations may or may not be provided in an order that is consistent with a causal ordering of the DAG associated to the SCM. This function returns the list of structural equations ordered in a manner consistent with a causal ordering of the associated DAG.

Returns *list* – The structural equations making up the SCM, causally ordered.

check_no_cycles(*atol=1e-06*)

Checks that the SCM defined is not cyclic.

Note that this function is essentially a wrapper around static method DirectedAcyclicGraph.validate_dag_adjacency_matrix, which relies on characterisation of acyclicity by the nilpotence of the adjacency matrix.

As eigenvalues are only ever computed approximately, it is theoretically possible than an eigenvalue is computed as very small but non-zero even though it is actually zero hence the use of a tolerance.

Parameters **atol** (*float, optional*) – The absolute tolerance used to check that the eigenvalues are all equal to 0 (default is 10^{-6}).

Raises ***CyclicityWarning*** – If the SCM is (highly likely to be) cyclic.

perform_intervention(*new_structural_equation*)
Performs an intervention on the SCM.

Performs an intervention on the SCM by replacing one of its constitutive structural equations by a new structural equation.

Parameters **new_structural_equation** (*StructuralEquation*) – The new structural equation.

Returns *StructuralCausalModel* – The post-intervention SCM.

Raises

- ***InvalidIntervention*** – If the new structural equation does not correspond to an existing one i.e. it has X_{i_0} on its left-hand side but there is no X_{i_0} variable in the SCM.
- ***CyclicityWarning*** – If the post-intervention SCM is (highly likely to be) cyclic.

1.1.10 StructuralCausalModels.structural_equation module

class StructuralCausalModels.structural_equation.**StructuralEquation**(*index_lhs*, *indices_rhs*, *exogenous_variable*, *function*)

Bases: object

A class to represent structural equations.

Structural Equations are assignments of the sort

$$X_i := g((X_j)_{j \in J}, U_i),$$

where U_i is an exogenous (random) variable.

Parameters

- **index_lhs** (*int*) – The index of the structural variable on the left-hand side of the structural equation (the "i").
- **indices_rhs** (*list*) – The indices of the structural variables on the right-hand side of the structural equation (the "j's in J").
- **exogenous_variable** (*scipy.stats.rv_continuous or scipy.stats.rv_discrete*) – The exogenous variable (the "U_i").
- **function** (*callable*) – The function defining the functional form of the assignment in the structural equation (the "g").

generate_data(*data*)
Generates samples from a structural equation.

Parameters **data** (*pandas.DataFrame*) – A dataframe containing data at least for the structural variables on the right-hand side of the structural equation.

If it contains data for the structural variable on the left-hand side of the structural equation, that data will be overwritten.

Returns *pandas.DataFrame* – The samples.

1.1.11 Module contents

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

S

INDEX

T

U

V