lgbn

Elias Hernandis

Jun 05, 2022

CONTENTS:

CHAPTER

ONE

API REFERENCE

1.1 Estimators

class lgbn.estimators.**GreedyEquivalentSearch**(*score=None*, *max_iter=1000000*, *eps=1e-09*)

Greedy Equivalent Search structure learning algorithm.

The Greedy Equivalent algorithm learns the structure of a Bayesian network that maximizes the given score. The search procedure starts with an empty graph. Edges are added until no more increase the score and then removed until no further operation increases the score. Equality of operations and thus networks is defined by equivalence classes. An equivalence class contains all networks which have the same edges regardless of orientation. This algorithm is reasonably fast when used with a decomposable score which can be cached.

See[3] for a detailed description of the Greedy Equivalent Search algorithm.

Note: This implementation requires a decomposable score, although there exist other implementations that work with non-decomposable scores.

References

get_params(*deep=True*)

Get parameters for this estimator.

Parameters
deep (*bool, default=True*) – If True, will return the parameters for this estimator and contained subobjects that are estimators.

Returns
params – Parameter names mapped to their values.

Return type
dict

search()

Search the space of posible models for the one that maximizes the score of this estimator.

set_params(***kwargs*)

Set the parameters of this estimator.

[3] D. M. Chickering, "Optimal Structure Identification With Greedy Search," Journal of Machine Learning Research, vol. 3, no. Nov 2002, p. 48, Nov. 2002.

The method works on simple estimators as well as on nested objects (such as `Pipeline`). The latter have parameters of the form `<component>__<parameter>` so that it's possible to update each component of a nested object.

Parameters
****params** (*dict*) – Estimator parameters.

Returns
self – Estimator instance.

Return type
estimator instance

`class lgbn.estimators.GreedyHillClimbing(score=None, start_net=None, max_iter=1000000, eps=1e-09, random_state=None)`

Greedy Hill Climbing structure search algorithm.

The Greedy Hill Climbing algorithm learns the structure of a Bayesian network that maximizes the given score. The search procedure starts with an initial network, which defaults to a fully disconnected network. Edges are added, removed or have their direction reversed one at a time until no more modifications increase the overall score of the network. This algorithm is reasonably fast when used with a decomposable score which can be cached.

See p. 40 in[2] for a detailed description of the Greedy Hill Climbing algorithm. The source refers to Greedy Hill Climbing as Max-Min Hill Climbing.

Note: This implementation requires a decomposable score, although there exist other implementations that work with non-decomposable scores.

References

`get_params(deep=True)`

Get parameters for this estimator.

Parameters
deep (*bool, default=True*) – If True, will return the parameters for this estimator and contained subobjects that are estimators.

Returns
params – Parameter names mapped to their values.

Return type
dict

`search()`

Search the space of posible models for the one that maximizes the score of this estimator.

`set_params(**kwargs)`

Set the parameters of this estimator.

The method works on simple estimators as well as on nested objects (such as `Pipeline`). The latter have parameters of the form `<component>__<parameter>` so that it's possible to update each component of a nested object.

[2] I. Tsamardinos, L. E. Brown, and C. F. Aliferis, "The max-min hill-climbing Bayesian network structure learning algorithm," Mach Learn, vol. 65, no. 1, pp. 31–78, Oct. 2006, doi: 10.1007/s10994-006-6889-7.

Parameters
****params** (*dict*) – Estimator parameters.

Returns
self – Estimator instance.

Return type
estimator instance

class lgbn.estimators.**K2Search**(*score=None*, *ordering=None*, *eps=1e-09*)

K2 structure learning algorithm.

The K2 algorithm learns the structure of a Bayesian network that maximizes the given score. The search procedure is guided by a given topological ordering of the network. In that ordering, if node x comes before node y, then node y can never be a parent of node x. This vastly reduces the search space resulting in a significant speedup, even without using caching.

See[1] for a detailed description of the K2 algorithm.

Note: This implementation requires a decomposable score, although there exist other implementations that work with non-decomposable scores.

References

get_params(*deep=True*)

Get parameters for this estimator.

Parameters
deep (*bool, default=True*) – If True, will return the parameters for this estimator and contained subobjects that are estimators.

Returns
params – Parameter names mapped to their values.

Return type
dict

search()

Search the space of posible models for the one that maximizes the score of this estimator.

set_params(***kwargs*)

Set the parameters of this estimator.

The method works on simple estimators as well as on nested objects (such as `Pipeline`). The latter have parameters of the form `<component>__<parameter>` so that it's possible to update each component of a nested object.

Parameters
****params** (*dict*) – Estimator parameters.

Returns
self – Estimator instance.

Return type
estimator instance

[1] G. F. Cooper and E. Herskovits, "A Bayesian method for the induction of probabilistic networks from data," Mach Learn, vol. 9, no. 4, pp. 309–347, Oct. 1992, doi: 10.1007/BF00994110.

class lgbn.estimators.**ScoreSearchEstimator**(*score=None*, *eps=1e-09*)

A structure estimator using score-based search.

Note: This class is just a general interface, it cannot actually be used.

See also:

K2Search, *GreedyHillClimbing*, *GreedyEquivalentSearch*

eps: float = 1e-09

Tolerance for equality testing of numeric values. Two values a and b are equal if `abs(a - b) < eps`.

fit(*data*)

Fit the model to the given data.

Parameters
data (*pandas.DataFrame*) – A DataFrame with one row per observation and one column per variable. Column names will be used for node identifiers in the resulting model.

Return type
A fitted estimator (self).

get_params(*deep=True*)

Get parameters for this estimator.

Parameters
deep (*bool, default=True*) – If True, will return the parameters for this estimator and contained subobjects that are estimators.

Returns
params – Parameter names mapped to their values.

Return type
dict

model: *BayesianNetwork* = None

The model resulting from the estimation.

property score

Score instance to use for scoring networks in the search procedure.

search() → *BayesianNetwork*

Search the space of posible models for the one that maximizes the score of this estimator.

set_params(***kwargs*)

Set the parameters of this estimator.

The method works on simple estimators as well as on nested objects (such as `Pipeline`). The latter have parameters of the form `<component>__<parameter>` so that it's possible to update each component of a nested object.

Parameters
****params** (*dict*) – Estimator parameters.

Returns
self – Estimator instance.

Return type
estimator instance

1.2 Models

class lgbn.models.**BayesianNetwork**

A Bayesian Network.

A Bayesian network is a directed acyclic graph where each node is a random variable with a distribution conditional on the parent nodes.

add_cpd(*cpd:* CPD) → None

Add a conditional probability distribution to the network.

Note: Also adds the node and edges to the underlying *networkx.DiGraph*. If there already is a conditional probability distribution it is replaced and new edges are added, but previous edges not present in the new distribution will not be removed.

Parameters

cpd (CPD) – A conditional probability distribution

apply_op(*op*)

Modify the network according to the given operation.

An operation is an edge together with an action (add edge, remove edge, flip edge).

Parameters

op – A tuple `(action, (u, v))` where action is one of +, - or F and (u, v) is an edge.

Raises

- **NotImplementedError** – If the given action is not supported.
- **ValueError** – If the edge is not in the network and operation requires editing it. (This is actually raised by *networkx.DiGraph.remove_edge*.)

cpd_class

The class used to instantiated CPDs when loading from a dict via *.from_dict(data)*.

alias of *CPD*

cpds

A dictionary mapping node identifiers to Conditional Probability Distributions.

alias of `Dict[Any, CPD]`

classmethod from_dict(*data*)

Load Bayesian network from a dict.

Parameters

data (*Sequence[Dict[str, Any]]*) – A list of dictionaries corresponding to CPDs. See *CPD.to_dict* for more information on the format.

See also:

to_dict, *CPD.from_dict*

Notes

This method expects the data to be sorted by node in topological order, so that a node always comes before its children. This is the way *to_dict* generates dictionaries.

to_dict()

Serializes the Bayesian network into a list of dictionaries.

Each dictionary is the result of serializing a CPD via *CPD.to_dict*.

update_cpds_from_structure() → None

Updates the *parents* attribute in each CPD to match the current graph structure.

Tip: Use this method after updating the network structure (e.g. via learning).

class lgbn.models.**CPD**(*node: Any*, *parents: Optional[tuple[Any]] = None*)

A conditional probability distribution for a node.which also references the node's parents.s

In a *BayesianNetwork* the probability distributions of the nodes are specified via CPDs or Conditional Probability distributions. These classes represent the probability distributions of the random variables in the nodes of a Bayesian network, which in general are conditioned on the parent nodes.

Note: This is a base class which cannot actually be used.

See also:

LinearGaussianCPD
A conditional probability distribution for linear Gaussian Bayesian networks.

classmethod from_dict(*data: Dict[str, Any]*)

Loads this conditional probability distribution from a dict.

to_dict() → Dict[str, Any]

Serializes this conditional probability distribution into a dict.

class lgbn.models.**LinearGaussianBayesianNetwork**

A Bayesian network where every node has a Gaussian distribution, where the mean of each node is a linear combination of its parents plus a bias factor and the standard deviations of the nodes are independent.

The joint distribution of these networks also a Gaussian distribution, the parameters of which can be obtained via the *to_joint_gaussian* method.

cpd_class

alias of *LinearGaussianCPD*

to_joint_gaussian()

Get the equivalent multivariate Gaussian distribution to this Bayesian network.

Returns a *scipy.stats.multivariate_normal* frozen random variable which has attributes *mean* (vector of means) and *cov* (covariance matrix).

Returns
A frozen random variable.

Return type
scipy.stats.multivariate_normal

Notes

Linear Gaussian Bayesian networks have a joint probability distribution that is also normal, and thus this method is well defined. See p. 252 of[1] and pp. 370-371 of[2].

References

class lgbn.models.**LinearGaussianCPD**(*node: Any*, *mean: Optional[float] = 0*, *var: Optional[float] = 1*, *parents: Optional[tuple[Any]] = None*, *weights: Optional[tuple[float]] = None*)

A linear Gaussian conditional probability distribution.

A linear Gaussian conditional probability distribution is normal distribution where the mean is a linear combination of the means of the parent nodes plus a bias term, i.e. if this node (X) has *parents* $U_1, \ldots, U_k$ then

$$p(X) = N(X \mid \mu + w_1\mu_{U_1} + \ldots + w_k\mu_{U_k}, \sigma^2),$$

where μ is specified via the *mean* parameter, σ^2 via the *var* parameter and $w_1, \ldots, w_k$ via the *weights* parameter.

mle(*data: DataFrame*)

Find maximum likelihood estimate for *mean*, *variance* and *weights* given the *data* and the dependencies on the *parents*.

Parameters
data (*pandas.DataFrame*) – A DataFrame with one row per observation and one column per variable.

Returns
A new conditional probability distribution where the parameters are set to the ML estimates.

Return type
LinearGaussianCPD

Notes

Maximum likelihood estimation of parameters is computed using the *sufficient statistics* approach described in section 17.2.4 of[1].

References

to_dict()

Serializes this conditional probability distribution into a dict.

[1] D. Koller and N. Friedman, Probabilistic graphical models: principles and techniques. Cambridge, MA: MIT Press, 2009.
[2] C. M. Bishop, Pattern recognition and machine learning. New York: Springer, 2006.

1.3 Scores

class lgbn.scores.**BICScore**(*data*)

The Bayesian Information Criterion score of a network is the LogLikScore of that network minus a regularization penalty proportional to the size of the data and the complexity of the network.

This score is decomposable and thus takes advantage of caching.

For a complete description of the BIC score see[1] section 18.3.5.

References

data: DataFrame

A Pandas DataFrame with one row per observation and one column per variable.

class lgbn.scores.**BaseScore**(*data: DataFrame*)

The base class for network scores.

A score is a class that takes an argument *data* on initialization and that implements a *score(network)* method that when given a network returns a real number.

This base class also stubs the implementation for a decomposable score, where the *.score(network)* function can be obtained by summing the *.score_fam(node, parent)* method over the nodes in a network.

data: DataFrame

A Pandas DataFrame with one row per observation and one column per variable.

score(*net:* BayesianNetwork)

A default implementation for decomposable scores where the score of a network is the sum of the scores of each family (i.e. the set of a node and its parents).

class lgbn.scores.**LogLikScore**(*data: DataFrame*)

The LogLik score of a network is the (natural) logarithm of the maximum likelihood of the data given the network as estimated by net.mle().

This score is decomposable and thus takes advantage of caching.

See section 18.3.1 of[Page 8, 1] for a more in-depth discussion of the log likelihood score.

data: DataFrame

A Pandas DataFrame with one row per observation and one column per variable.

[1] D. Koller and N. Friedman, Probabilistic graphical models: principles and techniques. Cambridge, MA: MIT Press, 2009.

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

INDEX

T

U