pyCausalFS：A Python Library of Causality-based Feature Selection for Causal Structure Learning and Classification

Public domain version 1.0 for Python

Release date: 02-06-2020

(Version 1.0)

References for citation:

- Kui Yu, Xianjie Guo, Zhaolong Ling, Lin Liu, Jiuyong Li, and Xindong Wu. Causality-based Feature Selection: Methods and Evaluations. arXiv:1911.07147 [cs.LG], 2019.

- Kui Yu, Lin Liu, and Jiuyong Li. A Unified View of Causal and Non-causal Feature Selection. arXiv:1802.05844 [cs.AI], 2018.

Authors:

- Wentao Hu (huwentao2013@foxmail.com)
- Mingzhu Cai(caimz199708@163.com)
- Yajing Yang(yyj13865934683@163.com)
- Kui Yu (yukui@hfut.edu.cn)

Hefei University of Technology
485 Danxia Road, Shushan District, Hefei, 230601, China

Table of Contents

I. Library overview

I.1. Introduction

The pyCausalFS library is written in Python. In the library, all algorithms and files are located at the "*pyCausalFS*" folder. The hierarchy of the pyCausalFS folder is as shown in Figure 1. The "*pyCausalFS*" folder consists of three subfolders: "*CBD*" [Constraint-based Markov Blanket or Parents and Children (MB/PC) learning algorithm for both continuous and discrete data using Fisher Z-test and χ^2 test respectively], "*SDD*" [Score-based MB/PC learning algorithms (only support discrete data)], and "*LSL*"[Local causal structure learning algorithm for both continuous and discrete data using Fisher Z-test and χ^2 test respectively]. For discrete data, the χ^2 test is employed while for continuous data, the Fisher Z-test is used.

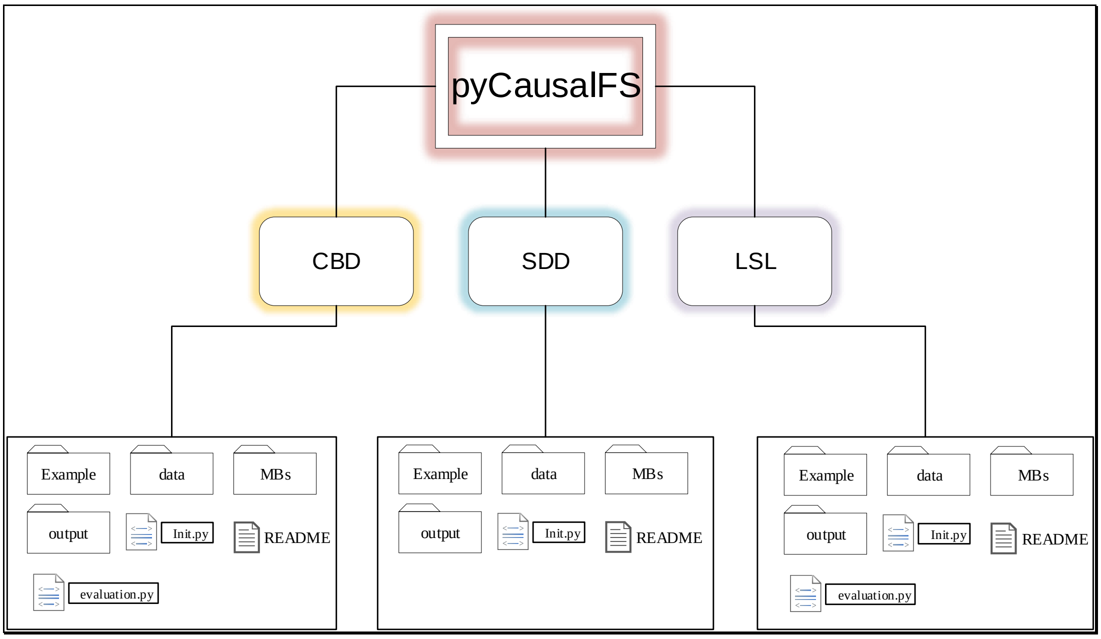


Figure 1 Hierarchy of the pyCausalFS folder in the pyCausalFS library

In Figure 1, all algorithms are kept in the "*MBs*" folder, such as IAMB, STMB and HITON-MB. Some python function modules which are frequently used in the PC/MB algorithms are stored in the "*/MBs/common*" folder. The "*Example*" folder keeps running examples of how to run *an* algorithm in the library.

The datasets are stored in the "*data*" folder for algorithm testing. For example, the file, named "*Child_s500_v1.csv*", is a training dataset for testing a MB, or PC, or local causal structure learning algorithm, where "Child" denotes that the dataset is generated from the benchmark Bayesian network "Child" and "s500" denotes that the sample size is 500. In addition, the format of the dataset for algorithm testing must be the ".csv" format. Otherwise the algorithms in the library cannot be correctly executed.

"*Child_graph.txt*" is the true DAG adjacency matrix of the benchmark Child BN.

"*evaluation.py*" is a Python function which includes all evaluation metrics for algorithm evaluating, such as Precision, Recall, F1, Distance, Running time (in seconds) and the number of conditional independence tests.

The evaluation results of the PC/MB learning algorithms and local causal structure learning algorithms are kept in the "*output*" folder.

I.2. Architecture of the Library

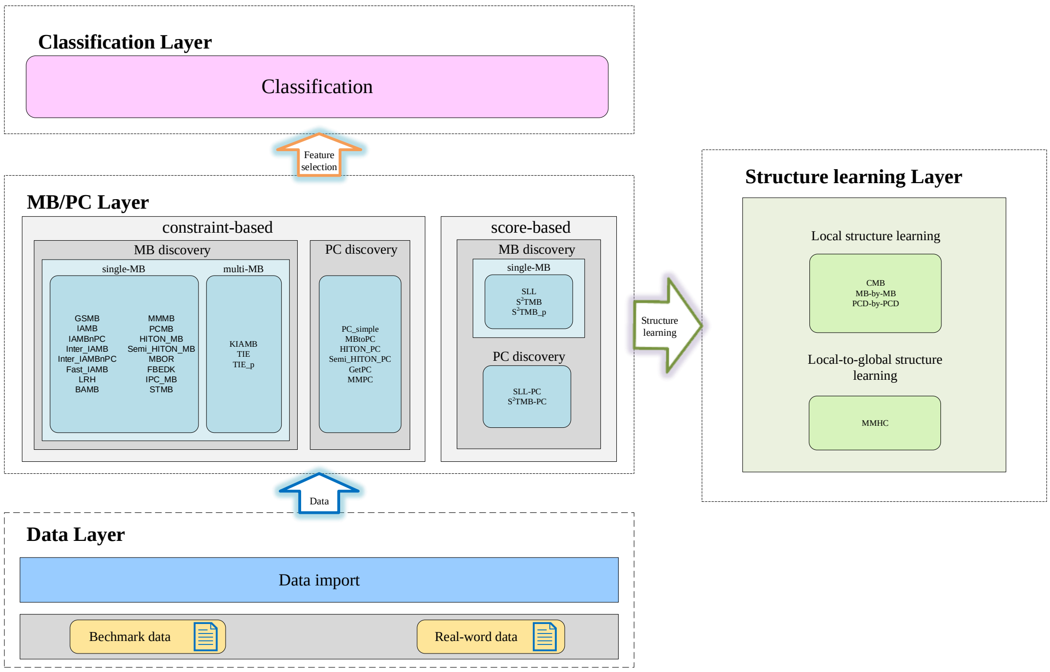


Figure 2 Architecture of the pyCausalFS library

As shown in Figure 2, the architecture of the pyCausalFS library consists of four modules, that is, data layer, MB/PC layer, structure learning layer, and classification layer. The four modules are designed independently. This makes that pyCausalFS is simple, easy to implement, and extendable flexibly. One can easily add a new algorithm to pyCausalFS and share it through pyCausalFS without modifying the other modules.

In the MB/PC layer, pyCausalFS implements 30 representative causality-based feature selection methods. Specifically, it consists of 25 methods using conditional independence tests (16 single MB learning algorithms, 3 multiple MB learning algorithms, and 6 PC learning algorithms), and 5 score-based approaches. Furthermore, by applying the MB and PC learning algorithms in the MB/PC layer, using the pyCausalFS library, users can easily design different local and global structure learning methods. The structure learning layer includes 3 local BN structure learning algorithms and one local-to global BN learning algorithm. In the classification layer, users can use the learnt the MB or PC set for training a classifier (e.g. SVM, KNN and NB) for classification. Thus the pyCausalFS library not only supports feature selection for supervised classification, but also is able to do local and global BN structure learning.

II. Environment configuration

II.1. Description

This package runs under PyCharm software of the Windows system, user can also install other software which can run python projects. Moreover, the Python library of the pyCausalFS library required should be installed in advance.

II.2. Configuring the package

The pyCausalFS library has following essential dependencies, which are kept in "*venv*" folder:

- python 3.0 or higher
- pandas
- scipy
- numpy
- scikit-learn
- network
- matplotlib
- Pillow
- pyBN

III. Constraint-based MB/PC algorithms

III.1. Description

For the constraint-based MB/PC algorithms, there is a folder named *"CBD"*. The algorithms in the *"CBD"* folder are used to deal with both of discrete and continuous data using conditional independence tests. The "*example_MB*" and "*example_PC*" algorithms located in the *"/CBD/example"* folder are used to help users to learn how to run a MB/PC algorithm to find the MB/PC of the given target variables, while the "*evaluation_MB*" algorithm in the *"CBD"* folder is used to get indicators of target nodes. All algorithms as shown in Tables 1, 3, and 5.

III.2. Inputs and outputs of the example_MB algorithms

Table 1 Inputs and outputs of the example_MB algorithms

Inputs (five parameters or six parameters only for KIAMB and FBED^k):	1st input = algorithm name
	extra input = K
	2nd input = data
	3rd input = target variable index
	4th input = alpha
	5th input = is_discrete
Outputs (two outputs):	1st output = MB of the target node
	2nd output = Running time

Inputs (five or six parameters):

In the above input parameters, each parameter is inputted one by one. There are six parameters in algorithms KIAMB or FBED^k, while other algorithms only have five parameters. The extra input parameter K is set for KIAMB and FBED^k. Each parameter should be set correctly, otherwise the algorithms cannot be correctly executed. In the following, we give the detailed explanations of these input parameters.

1st input = algorithm name

The algorithms as shown in Table 2 can be considered as the 1st input parameter.

Table 2 MB learning algorithm

- GSMB – Grow/Shrink MB algorithm
- IAMB – Incremental Association-Based Markov Blanket (IAMB)
- KIAMB – KAIMB algorithm for multiple MB learning

- inter_IAMB – Inter-IAMB algorithm
- fast_IAMB – Fast-IAMB algorithm
- IAMBnPC – IAMBnPC algorithm
- interIAMBnPC – inter-IAMBnPC algorithm
- LRH – Lessen swamping, resist masking and highlight the true positives
- BAMB – Balanced Markov blanket discovery (BAMB) algorithm
- FBEDk – Forward-Backward selection with Early Dropping algorithm
- MMMB – Min-Max Markov Blanket (MMMB) algorithm
- PCMB – Parents and children based MB (PCMB) algorithm
- HITON_MB – HITON_MB algorithm
- Semi_HITON_MB - Semi_HITON_MB algorithm
- MBOR – MB search using the OR condition
- IPCMB – Iterative Parent-Child based search of MB (IPCMB) algorithm
- STMB – Simultaneous MB discovery (STMB) algorithm
- TIE – Target Information Equivalence algorithm for multiple MB discovery (using Criterion Independence to verify Markov boundaries)
- TIE_p – Target Information Equivalence algorithm for multiple MB discovery (using Criterion Predictivity (Classifier)) to verify Markov boundaries (Naive Bayes classifier employed in the TIE_p algorithm)

extra input = K (K denotes the tradeoff between greediness and randomness for KIAMB, while for $FBED^k$ K denotes the number of additional runs)

For KIMB, the difference between KIAMB and IAMB is that KIAMB allows the user to specify the trade-off between greediness and randomness in the MB learning through a randomization parameter $K \in [0,1]$. IAMB greedily adds to the currently selected candidate MB set of the target variable T, CMB(T), the variable with the highest association with T among all variables excluding CMB(T), while KIAMB adds to CMB(T) the variables with the highest associations with T in the CanMB set which is a random subset of CMB(T) with the size $\max(1,\lfloor(|CMB(T)|\cdot K)\rfloor)$. K specifies the trade-off between greediness and randomness in the MB search: if setting K = 1, KIAMB reduces to IAMB, while if taking K = 0, KIAMB is a completely random approach expected to discover all the MBs of T with a nonzero probability if running repeatedly for enough times.

For $FBED^k$, K denotes the number of additional runs. The parameter K defines a family of algorithms, such as $FBED^0$, $FBED^1$ and $FBED^\infty$. $FBED^0$ performs the first run until termination, $FBED^1$ performs one additional run and $FBED^\infty$ performs runs until no more variables can be selected.

2^{nd} input = data

For an input dataset, columns denote variables and rows represent data observations. The dataset can be continuous or discrete. There are some discrete benchmark BN datasets in the data folder for using.

3^{rd} input = target variable index

If the number of nodes in a BN network is n, the index value of a node is among 0 to n-1. When users learn the MB sets of all nodes in the network, this parameter is set to "all". When users learn the MB sets of several nodes in the network, the input target node index need to be separated by ",". For example, "0,5,9" means an algorithm will return the MBs of nodes 0, 5, and 9.

4^{th} input = alpha

The "alpha" denotes the significance level for conditional independence tests (e.g., χ^2 test and Fisher Z-test). The level of significance for hypothesis testing often is set 0.01 or 0.05.

5^{th} input = is_discrete

The "is_discrete" denotes whether an input dataset is discrete or continuous. We set "1" to denote a discrete dataset, and "0" to mean a continuous dataset.

Outputs (two outputs):

1^{st} output = MB of the target nodes

The output of MB of the target nodes learnt by a MB algorithm will be shown in the terminal and written to the "*mb.txt*" file in the "*output*" folder.

2^{nd} output = Running time

The running time of a MB algorithm will be shown in the terminal and written to the "*mb.txt*" file in the "*output*" folder.

The example of running IAMB are shown in Figures 3 to 5 as follows.

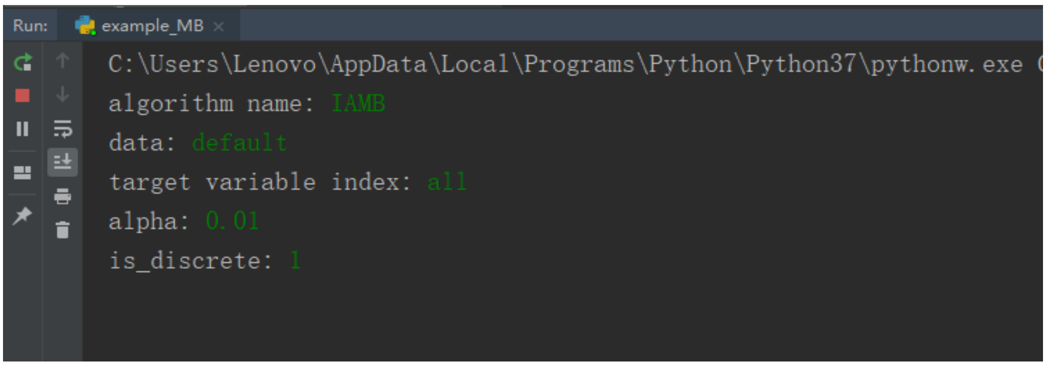


Figure 3 Before running the IAMB for learning MBs of all nodes

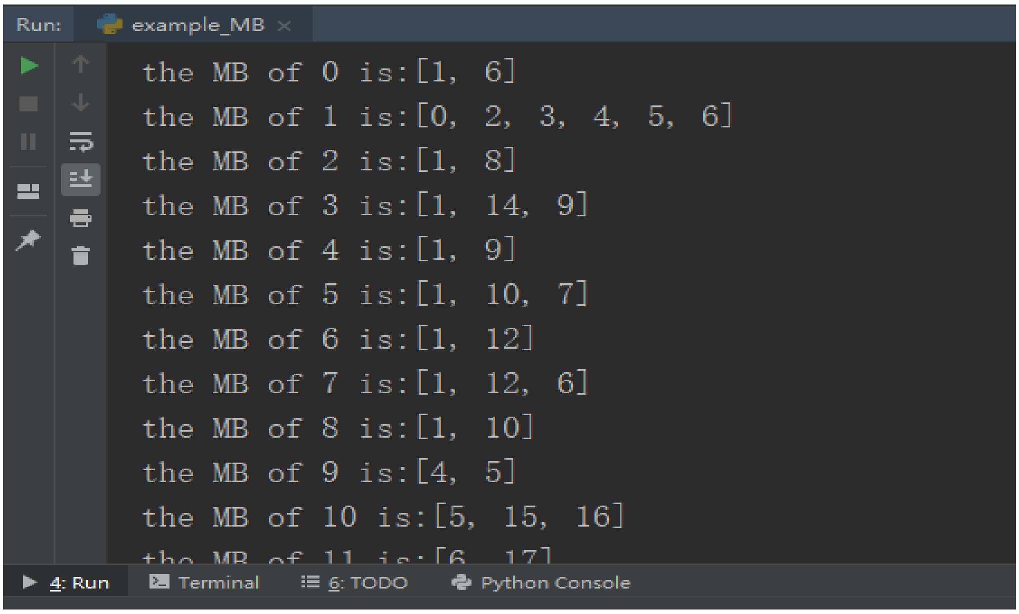


Figure 4 The MBs of the target nodes after running IAMB

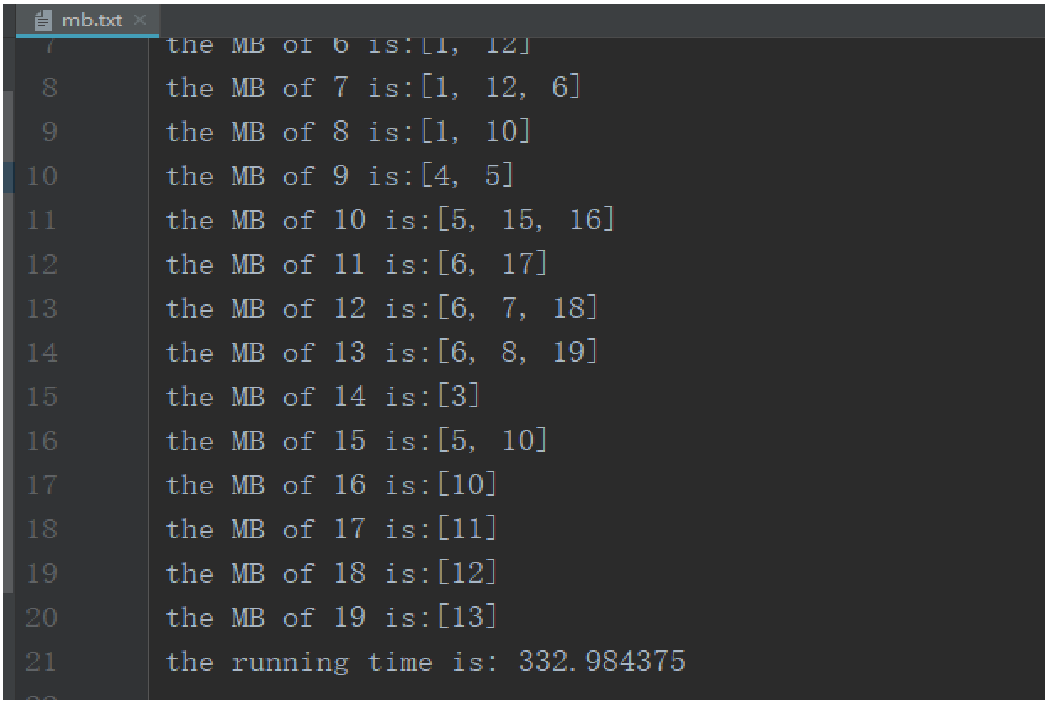


Figure 5 The “mb.txt” file after running IAMB

Other example of the MB algorithms:

GSMB algorithm:

algorithm name: GSMB data: default target variable index: all alpha: 0.01 is_discrete: 1

algorithm name: GSMB data: default target variable index: 1,5,9 alpha: 0.05 is_discrete: 1
algorithm name: GSMB data: ../data/Child_s500_v1.csv target variable index: 0 alpha: 0.05 is_discrete: 0

IAMB algorithm:

algorithm name: IAMB data: default target variable index: all alpha: 0.01 is_discrete: 1
algorithm name: IAMB data: default target variable index: 1,5,9 alpha: 0.05 is_discrete: 1
algorithm name: IAMB data: ../data/Child_s500_v1.csv target variable index: 0 alpha: 0.05 is_discrete: 0

KIAMB algorithm:

algorithm name: KIAMB k: 0.1 data: default target variable index: all alpha: 0.01 is_discrete: 1
algorithm name: KIAMB k: 0.1 data: default target variable index: 1,5,9 alpha: 0.05 is_discrete: 1
algorithm name: KIAMB k: 0.1 data: ../data/Child_s500_v1.csv

target variable index: 0 alpha: 0.05 is_discrete: 0

Inter_IAMB algorithm:

algorithm name: inter_IAMB data: default target variable index: all alpha: 0.01 is_discrete: 1
algorithm name: inter_IAMB data: default target variable index: 1,5,9 alpha: 0.05 is_discrete: 1
algorithm name: inter_IAMB data: ../data/Child_s500_v1.csv target variable index: 0 alpha: 0.05 is_discrete: 0

fast_IAMB algorithm:

algorithm name: fast_IAMB data: default target variable index: all alpha: 0.01 is_discrete: 1
algorithm name: fast_IAMB data: default target variable index: 1,5,9 alpha: 0.05 is_discrete: 1
algorithm name: fast_IAMB data: ../data/Child_s500_v1.csv target variable index: 0 alpha: 0.05 is_discrete: 0

IAMBnPC algorithm:

algorithm name: IAMBnPC data: default target variable index: all alpha: 0.01 is_discrete: 1

algorithm name: IAMBnPC data: default target variable index: 1,5,9 alpha: 0.05 is_discrete: 1
algorithm name: IAMBnPC data: ../data/Child_s500_v1.csv target variable index: 0 alpha: 0.05 is_discrete: 0

interIAMBnPC algorithm:

algorithm name: interIAMBnPC data: default target variable index: all alpha: 0.01 is_discrete: 1
algorithm name: interIAMBnPC data: default target variable index: 1,5,9 alpha: 0.05 is_discrete: 1
algorithm name: interIAMBnPC data: ../data/Child_s500_v1.csv target variable index: 0 alpha: 0.05 is_discrete: 0

MMMB algorithm:

algorithm name: MMMB data: default target variable index: all alpha: 0.01 is_discrete: 1
algorithm name: MMMB data: default target variable index: 1,5,9 alpha: 0.05 is_discrete: 1
algorithm name: MMMB data: ../data/Child_s500_v1.csv target variable index: 0 alpha: 0.05 is_discrete: 0

PCMB algorithm:

algorithm name: PCMB data: default target variable index: all alpha: 0.01 is_discrete: 1
algorithm name: PCMB data: default target variable index: 1,5,9 alpha: 0.05 is_discrete: 1
algorithm name: PCMB data: ../data/Child_s500_v1.csv target variable index: 0 alpha: 0.05 is_discrete: 0

IPCMB algorithm:

algorithm name: IPCMB data: default target variable index: all alpha: 0.01 is_discrete: 1
algorithm name: IPCMB data: default target variable index: 1,5,9 alpha: 0.05 is_discrete: 1
algorithm name: IPCMB data: ../data/Child_s500_v1.csv target variable index: 0 alpha: 0.05 is_discrete: 0

HITON_MB algorithm:

algorithm name: HITON_MB data: default target variable index: all alpha: 0.01 is_discrete: 1
algorithm name: HITON_MB data: default target variable index: 1,5,9 alpha: 0.05

is_discrete: 1
algorithm name: HITON_MB data: ../data/Child_s500_v1.csv target variable index: 0 alpha: 0.05 is_discrete: 0

Semi_HITON_MB algorithm:

algorithm name: semi_HITON_MB data: default target variable index: all alpha: 0.01 is_discrete: 1
algorithm name: semi_HITON_MB data: default target variable index: 1,5,9 alpha: 0.05 is_discrete: 1
algorithm name: semi_HITON_MB data: ../data/Child_s500_v1.csv target variable index: 0 alpha: 0.05 is_discrete: 0

STMB algorithm:

algorithm name: STMB data: default target variable index: all alpha: 0.01 is_discrete: 1
algorithm name: STMB data: default target variable index: 1,5,9 alpha: 0.05 is_discrete: 1
algorithm name: STMB data: ../data/Child_s500_v1.csv target variable index: 0 alpha: 0.05 is_discrete: 0

BAMB algorithm:

algorithm name: BAMB data: default

target variable index: all alpha: 0.01 is_discrete: 1
algorithm name: BAMB data: default target variable index: 1,5,9 alpha: 0.05 is_discrete: 1
algorithm name: BAMB data: ../data/Child_s500_v1.csv target variable index: 0 alpha: 0.05 is_discrete: 0

FBEDk algorithm:

algorithm name: FBEDk k: 1 data: default target variable index: all alpha: 0.01 is_discrete: 1
algorithm name: FBEDk k: 3 data: default target variable index: 1,5,9 alpha: 0.05 is_discrete: 1
algorithm name: FBEDk k: 10 data: ../data/Child_s500_v1.csv target variable index: 0 alpha: 0.05 is_discrete: 0

MBOR algorithm:

algorithm name: MBOR data: default target variable index: all alpha: 0.01 is_discrete: 1
algorithm name: MBOR data: default target variable index: 1,5,9 alpha: 0.05 is_discrete: 1

algorithm name: MBOR data: ../data/Child_s500_v1.csv target variable index: 0 alpha: 0.05 is_discrete: 0

LRH algorithm:

algorithm name: LRH data: default target variable index: all alpha: 0.01 is_discrete: 1
algorithm name: LRH data: default target variable index: 1,5,9 alpha: 0.05 is_discrete: 1
algorithm name: LRH data: ../data/Child_s500_v1.csv target variable index: 0 alpha: 0.05 is_discrete: 0

TIE algorithm:

algorithm name: TIE data: default target variable index: all alpha: 0.01 is_discrete: 1
algorithm name: TIE data: default target variable index: 1,5,9 alpha: 0.05 is_discrete: 1
algorithm name: TIE data: ../data/Child_s500_v1.csv target variable index: 0 alpha: 0.05 is_discrete: 0

TIE_p algorithm:

algorithm name: TIE_p data: default target variable index: all

alpha: 0.01 is_discrete: 1
algorithm name: TIE_p data: default target variable index: 1,5,9 alpha: 0.05 is_discrete: 1
algorithm name: TIE_p data: ../data/Child_s500_v1.csv target variable index: 0 alpha: 0.05 is_discrete: 0

III.3. Inputs and outputs of the example_PC Algorithms

Table 3 Inputs and outputs of the example_PC algorithms

Inputs (five parameters):	1st input = algorithm name
	2nd input = data
	3rd input = Target variable index
	4th input = alpha
	5th input =is_discrete
Outputs (two outputs):	1st output = PC of each target node
	2nd output = Running time

Inputs (five parameters):

Each parameter should be inputted correctly, otherwise the program cannot be correctly executed. In the following, we give the detailed explanations of these input parameters.

1st input = Algorithm name

The algorithms as shown in Table 4 can be considered as the 1st input parameter.

Table 4 PC learning algorithm

- MBtoPC – MBtoPC algorithm
- pc_simple – PC simple algorithm
- HITON_PC – HITON_PC algorithm
- MMPC – Max-Min Parents and Children algorithm
- getPC – GetPC algorithm
- semi_HITON_PC – Semi_HITON_PC algorithm

2nd input = data

For an input dataset, columns denote variables and rows represent data observations. The dataset can be continuous or discrete. There are some benchmark BN datasets in the data folder for using.

3rd input = Target variable index
If the number of nodes in a BN network is n, the index value of a node is among 0 to n-1. When users learn the MB sets of all nodes in the network, this parameter is set to "all". When users learn the MB sets of several nodes in the network, the input target node index need to be separated by ",". For example, "0,5,9" means an algorithm will return the MBs of nodes 0, 5, and 9.

4th input = alpha
The “alpha” denotes the significance level for conditional independence tests (e.g., χ^2 test and Fisher Z-test). The level of significance for hypothesis testing often is set 0.01 or 0.05.

5th input = is_discrete
The “is_discrete” denotes whether an input dataset is discrete or continuous. We set “1” to denote a discrete dataset, and “0” to mean a continuous dataset.

Outputs (two outputs):

1st output = PC of the target nodes
The output of PC of the target nodes learnt by a MB algorithm will be shown in the terminal and written to the “*pc.txt*” file in the “*output*” folder.

2nd output = Running time
The running time of a PC algorithm will be shown in the terminal and written to the “*pc.txt*” file in the “*output*” folder.

The example of running MBtoPC algorithm are shown in Figures 6 to 8 as follows.

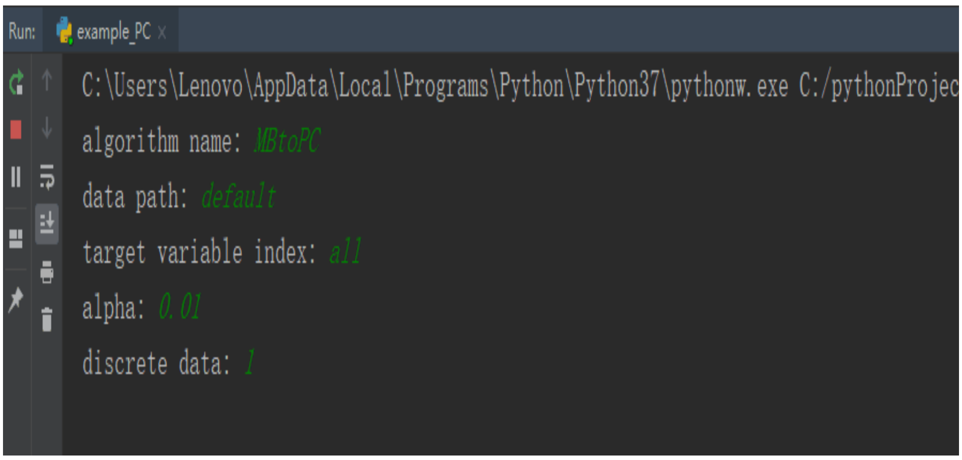


Figure 6 Before running the MBtoPC algorithm for learning PCs of all nodes

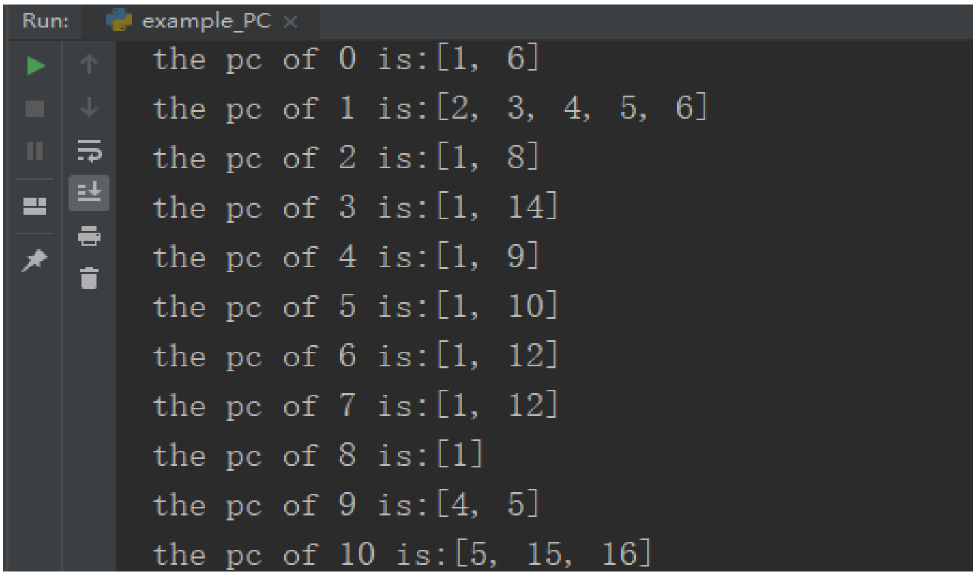


Figure 7 The MBs of the target nodes after running MBtoPC algorithm

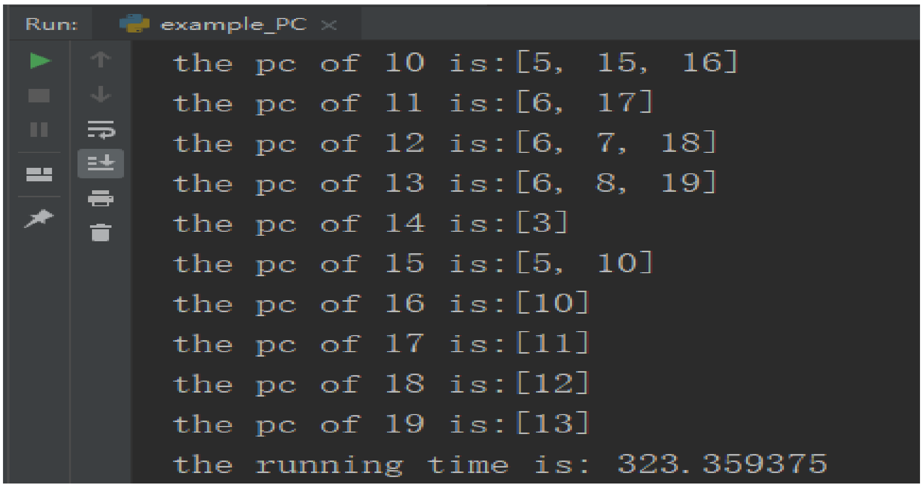


Figure 8 The “mb.txt” file after running MBtoPC algorithm

Other example of the PC algorithms:

MBtoPC algorithm:

algorithm name: MBtoPC data: default target variable index: all alpha: 0.01 is_discrete: 1
algorithm name: MBtoPC data: default target variable index: 1,5,9

alpha: 0.05 is_discrete: 1
algorithm name: MBtoPC data: ../data/Child_s500_v1.csv target variable index: 0 alpha: 0.05 is_discrete: 0

pc_simple algorithm:

algorithm name: pc_simple data: default target variable index: all alpha: 0.01 is_discrete: 1
algorithm name: pc_simple data: default target variable index: 1,5,9 alpha: 0.05 is_discrete: 1
algorithm name: pc_simple data: ../data/Child_s500_v1.csv target variable index: 0 alpha: 0.05 is_discrete: 0

HITON_PC algorithm:

algorithm name: HITON_PC data: default target variable index: all alpha: 0.01 is_discrete: 1
algorithm name: HITON_PC data: default target variable index: 1,5,9 alpha: 0.05 is_discrete: 1
algorithm name: HITON_PC data: ../data/Child_s500_v1.csv target variable index: 0 alpha: 0.05 is_discrete: 0

MMPC algorithm:

algorithm name: MMPC data: default target variable index: all alpha: 0.01 is_discrete: 1
algorithm name: MMPC data: default target variable index: 1,5,9 alpha: 0.05 is_discrete: 1
algorithm name: MMPC data: ../data/Child_s500_v1.csv target variable index: 0 alpha: 0.05 is_discrete: 0

semi_HITON_PC algorithm:

algorithm name: semi_HITON_PC data: default target variable index: all alpha: 0.01 is_discrete: 1
algorithm name: semi_HITON_PC data: default target variable index: 1,5,9 alpha: 0.05 is_discrete: 1
algorithm name: semi_HITON_PC data: ../data/Child_s500_v1.csv target variable index: 0 alpha: 0.05 is_discrete: 0

III.4. Inputs and outputs of the evaluation_MB Algorithms

Table 5 Inputs and outputs of the evaluation_MB algorithms

Inputs (seven parameters or eight parameters only for KIAMB and FBED^k):	1st input = Algorithm name
	extra input= k
	2nd input = true DAG
	3rd input = data
	4th input = file number
	5th input = target variable index
	6th input = alpha
	7th input = is _discrete
Outputs (six outputs):	1st output = F1
	2nd output = Precision
	3rd output = Recall
	4th output = Distance
	5th input = ci_number
	6th input = time

Inputs ((seven parameters or eight parameters):

In the above input parameters, each parameter is inputted line by line. There are six parameters in algorithms KIAMB or FBED^k, while other algorithms only have five parameters. The extra input parameter K is set for KIAMB and FBED^k. Each parameter should be set correctly, otherwise the algorithms cannot be correctly executed. In the following, we give the detailed explanations of these input parameters.

1st input = Algorithm name

The algorithms as shown in Table 6 can be considered as the 1st input parameter.

Table 6 MB learning algorithm

- GSMB – Grow/Shrink MB algorithm
- IAMB – Incremental Association-Based Markov Blanket (IAMB)
- KIAMB – KAIMB algorithm for multiple MB learning
- inter_IAMB – Inter-IAMB algorithm
- fast_IAMB – Fast-IAMB algorithm
- IAMBnPC – IAMBnPC algorithm
- interIAMBnPC – interIAMBnPC algorithm
- LRH – Lessen swamping, resist masking and highlight the true positives
- BAMB – Balanced Markov blanket discovery (BAMB) algorithm
- FBEDk – Forward-Backward selection with Early Dropping algorithm
- MMMB – Min-Max Markov Blanket (MMMB) algorithm
- PCMB – Parents and children based MB (PCMB) algorithm
- HITON_MB – HITON_MB algorithm

- Semi_HITON_MB – Semi_HITON_MB algorithm
- MBOR – MB search using the OR condition
- IPCMB – Iterative Parent-Child based search of MB (IPCMB) algorithm
- STMB – Simultaneous MB discovery (STMB) algorithm

extra input = K (K denotes the tradeoff between greediness and randomness for KIAMB, while for $FBED^k$ K denotes the number of additional runs)

For KIMB, the difference between KIAMB and IAMB is that KIAMB allows the user to specify the trade-off between greediness and randomness in the MB learning through a randomization parameter $K \in [0,1]$. IAMB greedily adds to the currently selected candidate MB set of the target variable T, CMB(T), the variable with the highest association with T among all variables excluding CMB(T), while KIAMB adds to CMB(T) the variables with the highest associations with T in the CanMB set which is a random subset of CMB(T) with the size $\max(1,\lfloor(|CMB(T)|\cdot K)\rfloor)$. K specifies the trade-off between greediness and randomness in the MB search: if setting K = 1, KIAMB reduces to IAMB, while if taking K = 0, KIAMB is a completely random approach expected to discover all the MBs of T with a nonzero probability if running repeatedly for enough times.

For $FBED^k$, K denotes the number of additional runs. The parameter K defines a family of algorithms, such as $FBED^0$, $FBED^1$ and $FBED^\infty$. $FBED^0$ performs the first run until termination, $FBED^1$ performs one additional run and $FBED^\infty$ performs runs until no more variables can be selected.

2nd input = true DAG

The parameter of true DAG is the adjacency matrix of a true DAG of a Bayesian network and it is used for comparing the MB/PC of a node learned by an MB/PC algorithm with the true MB/PC of the node in the true DAG. Users should input the absolute or relative paths of the true graph and the format of graph should be the ".txt" format, such as "C:CBD\data\Child_graph.txt" or "..\data\Child_graph.txt". In addition, we set "default" as "..\data\Child_graph.txt". There is a benchmark BN graph in the data folder.

3rd input = data

The first thing worth noting is that the input parameter is different from the above program. The input of data denote that you should input the <u>incompletely</u> absolute or relative paths of the dataset, such as "C:CBD/data/Child_s500_v" or "../data/Child_s500_v". It different from above example_MB and example_PC program, because users probably need one or more dataset to evaluate one algorithm. The evaluation_MB program will be automatically spliced above input into a form such as "../data/Child_s500_v1.csv" or "C:CBD/data/Child_s500_v1.csv". In addition, we set "default" as " ../data/Child_s500_v" and it will be spliced into "../data/Child_s500_v1.csv". The format of dataset should be the ".csv" format,

otherwise the program cannot be correctly executed.
The dataset is used for training. Columns are variables and rows are observations. Support both continuous and discrete sample datasets. There are some benchmark BN datasets in the data folder for using.

4th input = file number
The "file number" denotes the number of datasets used in one evaluation. For example, if file number=10, it means that the evaluation will run on 10 different datasets to evaluate the performance of a MB algorithm. Moreover, these files must be of the same type and named sequentially, such as "../data/ Child_s500_v1.csv" to "../data/Child_s500_v9.csv".

5th input = target variable index
If the number of nodes in a BN network is n, the index value of a node is among 0 to n-1. When users learn the MB sets of all nodes in the network, this parameter is set to "all". When users learn the MB sets of several nodes in the network, the input target node index need to be separated by ",". For example, "0,5,9" means an algorithm will return the MBs of nodes 0, 5, and 9.

6th input = alpha
The "alpha" denotes the significance level for conditional independence tests (e.g., χ^2 test and Fisher Z-test). The level of significance for hypothesis testing often is set 0.01 or 0.05.

7thinput = is _discrete
The "is_discrete" denotes whether an input dataset is discrete or continuous. We set "1" to denote a discrete dataset, and "0" to mean a continuous dataset.

Outputs (six outputs):
For MB learning algorithms, the evaluation metrics include 1st output = F1, 2nd output = Precision, 3rd output = Recall, 4th output = Distance, 5th input = ci_number (number of independence tests(only for constraint-based MB learning algorithms)) and 6th input =Running time(in seconds) The metric results will be written to the "*output*" folder.

The example of evaluation of KIAMB algorithm are shown in Figure 9 to 11 as follows:

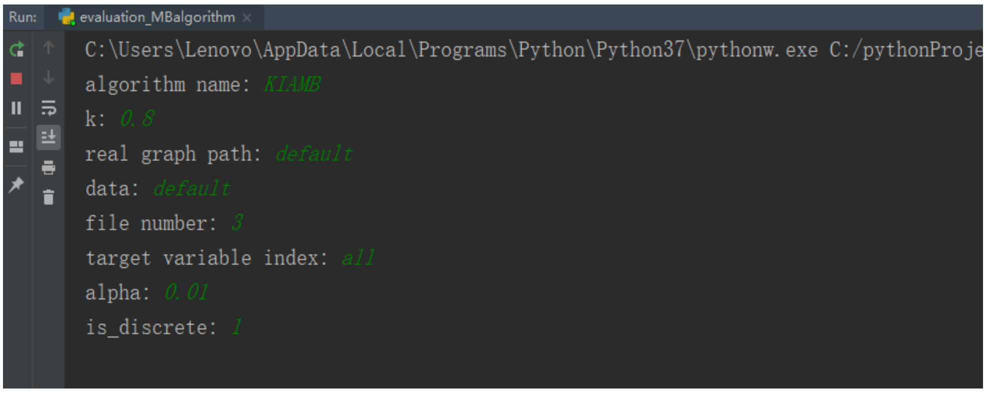


Figure 9 Before running the evaluation of KIAMB algorithm

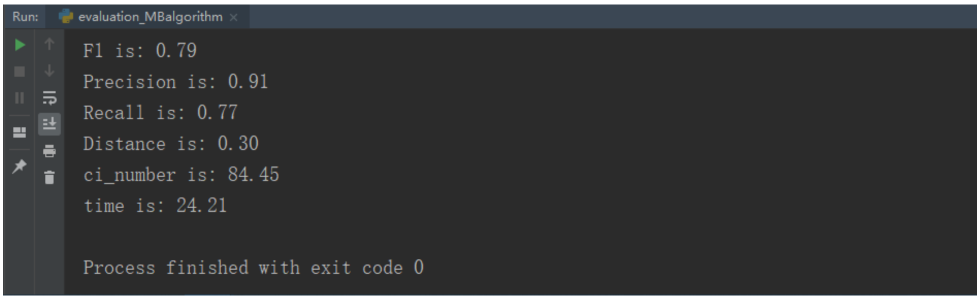


Figure 10 The MBs of the target nodes after running KIAMB algorithm

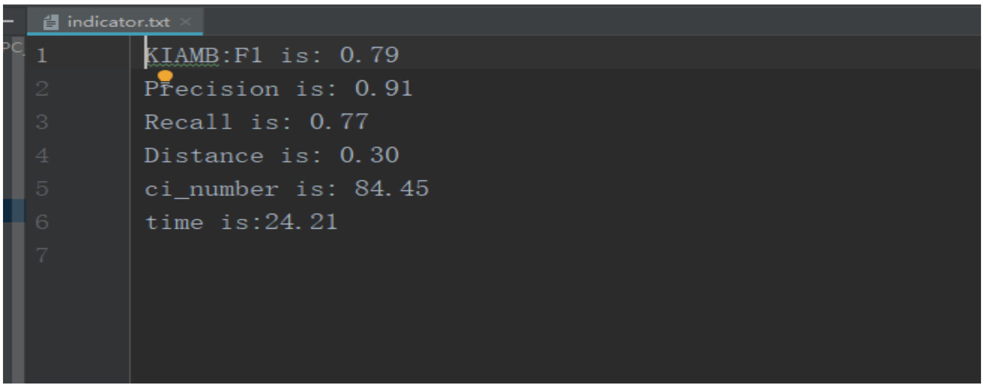


Figure 11 “indicator.txt” file after running KIAMB algorithm

Other example of the evaluation_MB algorithms:

GSMB algorithm:

algorithm name: MMMB real graph path: default data: default file number: 10 target variable index: all alpha: 0.01 is_discrete: 1

IAMB algorithm:

```
algorithm name:   IAMB
real graph path:   default
data:   ../data/Child_s500_v
file number: 10
target variable index:0,5,9
alpha: 0.05
is_discrete:   0
```

KIAMB algorithm:

```
algorithm name:   KIAMB
K: 0.8
real graph path:   default
data: default
file number: 10
target variable index: all
alpha: 0.01
is_discrete:   1
```

inter_IAMB algorithm:

```
algorithm name:   Inter_IAMB
real graph path:   default
data:   ../data/Child_s500_v
file number: 10
target variable index:0,5,9
alpha: 0.01
is_discrete:   1
```

fast_IAMB algorithm:

```
algorithm name:   fast_IAMB
real graph path:   default
data:   default
file number: 10
target variable index: all
alpha: 0.05
is_discrete:   0
```

IAMBnPC algorithm:

algorithm name: IAMBnPC
real graph path: default
data: ../data/Child_s500_v
file number: 10
target variable index:0,5,9
alpha: 0.01
is_discrete: 1

interIAMBnPC algorithm:

algorithm name: interIAMBnPC
real graph path: default
data: default
file number: 10
target variable index: all
alpha: 0.05
is_discrete: 0

LRH algorithm:

algorithm name: LRH
real graph path: default
data: ../data/Child_s500_v
file number: 10
target variable index:0,5,9
alpha: 0.01
is_discrete: 1

BAMB algorithm:

algorithm name: BAMB
real graph path: default
data: default
file number: 10
target variable index: all
alpha: 0.05
is_discrete: 0

FBEDk algorithm:

```
algorithm name:   FBEDk
K: 0.8
real graph path:   default
data:   ../data/Child_s500_v
file number: 10
target variable index: 0,5,9
alpha: 0.01
is_discrete:   1
```

MMMB algorithm:

```
algorithm name:   MMMB
real graph path:   default
data:   default
file number: 10
target variable index: all
alpha: 0.05
is_discrete:   0
```

PCMB algorithm:

```
algorithm name:   PCMB
real graph path:   default
data:   ../data/Child_s500_v
file number: 10
target variable index: 0,5,9
alpha: 0.01
is_discrete:   1
```

HITON_MB algorithm:

```
algorithm name:   HITON_MB
real graph path:   default
data:   default
file number: 10
target variable index: all
alpha: 0.05
is_discrete:   0
```

Semi_HITON_MB algorithm:

```
algorithm name:    Semi_HITON_MB
real graph path:    default
data:    ../data/Child_s500_v
file number: 10
target variable index: 0,5,9
alpha: 0.01
is_discrete:    1
```

MBOR algorithm:

```
algorithm name:    MBOR
real graph path:    default
data:    default
file number: 10
target variable index: all
alpha: 0.05
is_discrete:    0
```

IPCMB algorithm:

```
algorithm name:    IPCMB
real graph path:    default
data:    ../data/Child_s500_v
file number: 10
target variable index: 0,5,9
alpha: 0.01
is_discrete:    1
```

STMB algorithm:

```
algorithm name:    STMB
real graph path:    default
data:    default
file number: 10
target variable index: all
alpha: 0.05
is_discrete:    0
```

IV. Score-based MB/PC algorithms

IV.1. Description

The score-based project is located at the *"SDD"* folder, which can only be used to deal with discrete datasets. All algorithms as shown in Table 7 have the same number and types of input and output parameters.

IV.2. Inputs and outputs

Table 7 Inputs and outputs of the score-based algorithms

Inputs (three parameters):	1st input = algorithm name
	2nd input = data
	3rd input = target variable index
Outputs (two outputs):	1st output = the PC/MB node of each target node
	2nd output = Running time

Inputs (three parameters):

In the above input parameters, every parameter is inputted line by line. Each parameter should be set correctly, otherwise the program cannot be correctly executed. In the following, we give the detailed explanation of these input parameters.

1st input = algorithm name

The algorithms as shown in Table 8 can be considered as the 1st input parameter, such as SLL, S2TMB, and S2TMB_p.

Table 8 MB learning algorithm

- SLL – Score-based Local learning algorithm
- SLL_PC – SLL_PC algorithm
- S2TMB – Score-based Simultaneous Markov Blanket discovery algorithm
- S2TMB_PC – S^2TMB_PC algorithm
- S2TMB_p – Score-Based Simultaneous Markov Blanket+ algorithm (i.e. $S2TMB^+$)

2nd input = data

The dataset is used for training. Columns are variables and rows are observations. Support both continuous and discrete sample datasets. There are some benchmark BN datasets in the data folder for using.

3rd input = Target variable index

If the number of nodes in a BN network is n, the index value of a node is among 0 to n-1. When users learn the MB sets of all nodes in the network, this parameter is set to "all". When users learn the MB sets of several nodes in the network, the input target node index need to be separated by ",". For example, "0,5,9" means an algorithm will return the MBs of nodes 0, 5, and 9.

Outputs (two outputs):

1st output = MB/PC of the target nodes

The output of MB/PC of the target nodes learnt by a MB/PC algorithm will be shown in the terminal and written to the “*mb.txt*” file in the “*output*” folder.

2nd output = Running time

The running time of a MB algorithm will be shown in the terminal and written to the “*mb.txt*” file in the “*output*” folder.

The example of running SLL are shown in Figures 12 to 14 as follows:

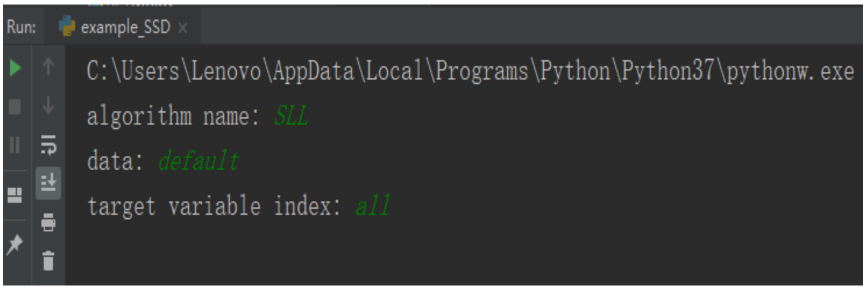


Figure 12 After running the SLL algorithm for learning MBs of all node

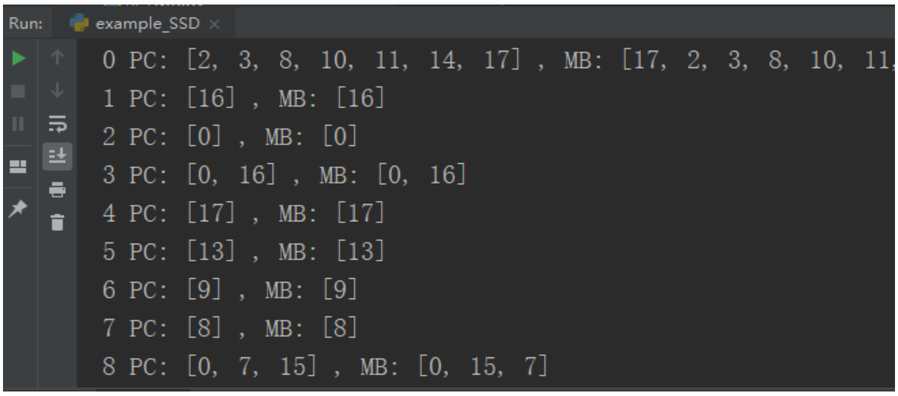
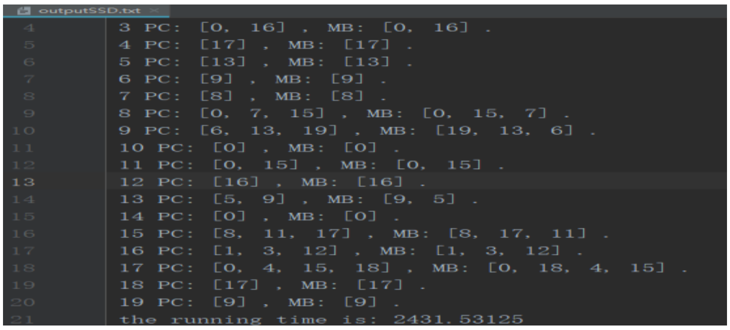


Figure 13 The MBs of the target nodes after running SLL algorithm



```
3 PC: [0, 16] , MB: [0, 16] .
4 PC: [17] , MB: [17] .
5 PC: [13] , MB: [13] .
6 PC: [9] , MB: [9] .
7 PC: [8] , MB: [8] .
8 PC: [0, 7, 15] , MB: [0, 15, 7] .
9 PC: [6, 13, 19] , MB: [19, 13, 6] .
10 PC: [0] , MB: [0] .
11 PC: [0, 15] , MB: [0, 15] .
12 PC: [16] , MB: [16] .
13 PC: [5, 9] , MB: [9, 5] .
14 PC: [0] , MB: [0] .
15 PC: [8, 11, 17] , MB: [8, 17, 11] .
16 PC: [1, 3, 12] , MB: [1, 3, 12] .
17 PC: [0, 4, 15, 18] , MB: [0, 18, 4, 15] .
18 PC: [17] , MB: [17] .
19 PC: [9] , MB: [9] .
the running time is: 2431.53125
```

Figure 14 "outputSSD.txt" written after running the SLL algorithm for learning MBs of all nodes

Other example of the score-based algorithms:

SLL_PC algorithm:

algorithm name: SLL_PC data: default target variable: 0,1,2
algorithm name: SLL_PC data: ../data/Child_s500_v1.csv target variable: 0,1,2

S2TMB algorithm:

algorithm name: S2TMB data: default target variable: 0,1,2
algorithm name: S2TMB data: ../data/Child_s500_v1.csv target variable: 0,1,2

S2TMB_PC algorithm:

algorithm name: S2TMB_PC data: default target variable: 0,1,2
algorithm name: S2TMB_PC data: ../data/Child_s500_v1.csv target variable: 0,1,2

S2TMB_p algorithm:

algorithm name: S2TMB_p data: default target variable: 0,1,2
algorithm name: S2TMB_p data: ../data/Child_s500_v1.csv target variable: 0,1,2

V. Local structure Learning algorithms

V.1. Description

The local structure learning structure project is located at the *"LSL"* folder, which can be used to deal with both discrete and continuous datasets. The "example_LSL" function in the *"/LSL/example"* folder is used to learn the MB of any target variable nodes, while "evaluation_LSL" function in the *"LSL"* folder is used to get indicators of any target nodes.

V.2. Inputs and outputs of example_LSL Algorithms

Table 9 Inputs and outputs of local structure learning algorithms

Inputs (three parameters):	1st input = algorithm name
	2nd input = data
	3rd input = target variable index
	4th input = significance level
	5th input = is_discrete
Outputs (two outputs):	1st output = the PC or MB node of each target node
	2nd output = Running time

Inputs (three parameters):

In the above input parameters, each parameter is inputted one by one. Each parameter should be inputted correctly, otherwise the algorithm cannot be correctly executed. In the following, we give the detailed explanations of these input parameters.

1st input = algorithm name

The algorithms as shown in Table 10 can be considered as the 1st input parameter.

Table 10 local structure learning algorithm

- PCDbyPCD – PCD-by-PCD algorithm
- MBbyMB – MB-by-MB algorithm
- CMB – Causal Markov Blanket algorithm

2nd input = data

For an input dataset, columns denote variables and rows represent data observations. The dataset can be continuous or discrete. There are some benchmark BN datasets in the data folder for using.

3rd input = Target variable index

If the number of nodes in a BN network is n, the index value of a node is among 0 to n-1. When users learn the MB sets of all nodes in the network, this parameter is set to "all". When users learn the MB sets of several nodes in the network, the input target node index need to be separated by ",". For example, "0,5,9" means an algorithm will return the MBs of nodes 0, 5, and 9.

4th input = alpha

The "alpha" denotes the significance level for conditional independence tests (e.g., χ^2 test and Fisher Z-test). The level of significance for hypothesis testing often is set 0.01 or 0.05.

5th input = is_discrete
The "is_discrete" denotes whether an input dataset is discrete or continuous. We set "1" to denote a discrete dataset, and "0" to mean a continuous dataset.

Outputs (two outputs):

1st output = MB of the target nodes
The output of MB of the target nodes learnt by a MB algorithm will be shown in the terminal and written to the "*mb.txt*" file in the "*output*" folder.

2nd output = Running time
The running time of a MB algorithm will be shown in the terminal and written to the "*mb.txt*" file in the "*output*" folder.

The example of running CMB algorithm are shown in Figures 15 to 17 as follows:

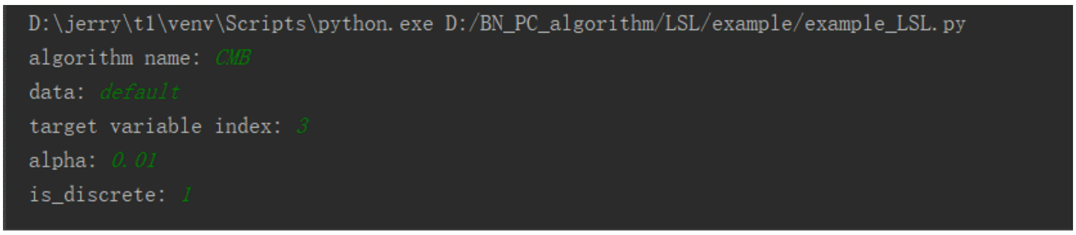


Figure 15 Before running the CMB algorithm for learning MBs of all node

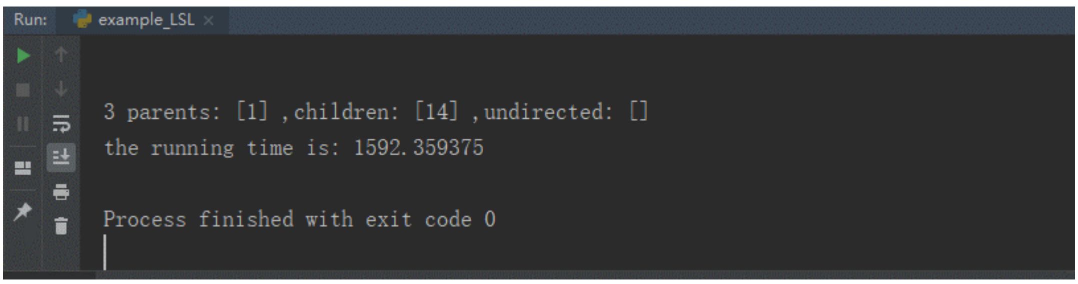


Figure 16 The MBs of the target nodes after running CMB algorithm

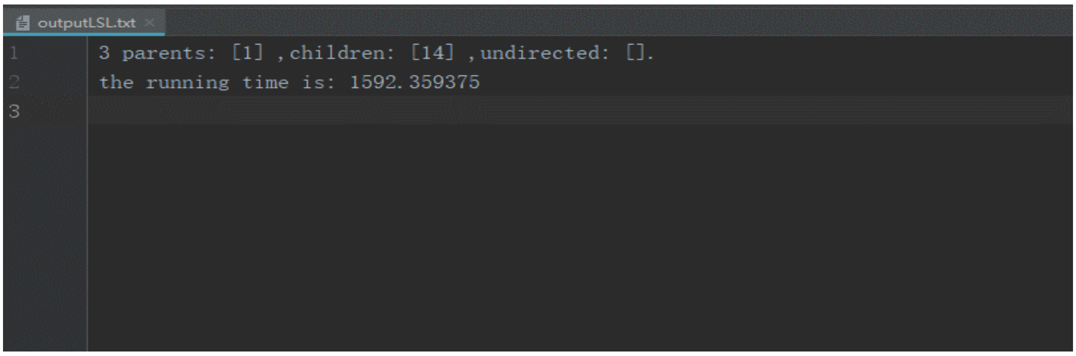


Figure 17 “outputLSL.txt” file after running CMB algorithm

Other example of the example_LSL algorithms:

PCD-by-PCD algorithm:

algorithm name: PCDbyPCD data: default target variable index: all alpha: 0.01 is_discrete: 1
algorithm name: PCDbyPCD data: ../data/Child_s500_v1.csv target variable index: 0,1,2 alpha: 0.05 is_discrete: 0

MB-by-MB algorithm:

algorithm name: MBbyMB data: default target variable index: all alpha: 0.01 is_discrete: 1
algorithm name: MBbyMB data: ../data/Child_s500_v1.csv target variable index: 0,1,2 alpha: 0.05 is_discrete: 0

CMB algorithm:

algorithm name: CMB data: default target variable index: all alpha: 0.01 is_discrete: 1

algorithm name: CMB data: ../data/Child_s500_v1.csv target variable index: 0,1,2 alpha: 0.05 is_discrete: 0

V.3. Inputs and outputs of evaluation_LSL Algorithms

Table 11 Inputs and outputs of evaluation_LSL algorithms

Inputs (seven parameters [k]):	1st input = Algorithm name
	2nd input = true DAG
	3rd input = data
	4th input = file number
	5th input = target variable index
	6th input = alpha
	7th input = is _discrete
Outputs (three outputs):	1st output = reverse
	2nd output = miss
	3rd output = extra

Inputs (seven parameters):

In the above input parameters, every parameter is inputted line by line. Each parameter should be inputted correctly, otherwise the algorithm cannot execute. In the following, we give the detailed explanation of these input parameters.

1st input = Algorithm name

The algorithms as shown in Table 11 can be considered as the 1st input parameter.

Table 12 local structure learning algorithm

● PCDbyPCD – PCD-by-PCD algorithm ● MBbyMB – MB-by-MB algorithm ● CMB – Causal Markov Blanket algorithm

2nd input = true DAG

The parameter of true DAG is the adjacency matrix of a true DAG of a Bayesian network and it is used for comparing the local structure around a node learned by a local structure learning algorithm with the local structure around the node in the true DAG. Users should input the absolute or relative paths of the true graph and the format of graph should be the ".txt" format, such as "C:CBD\data\Child_graph.txt" or "..\data\Child_graph.txt". In addition, we set "default" as "..\data\Child_graph.txt". There is a benchmark BN graph in the data folder.

3rd input = data

The first thing worth noting is that the input parameter is different from the above program. The input of data denote that you should input the incompletely absolute or relative paths of the dataset, such as “C:CBD/data/Child_s500_v” or “../data/Child_s500_v”. It different from above example_LSL program, because user probably need one or more dataset to evaluate one algorithm. The evaluation_MB program will be automatically splice above input into a form such as “../data/Child_s500_v1.csv” or “C:CBD/data/Child_s500_v1.csv”. In addition, we set “default” as “ ../data/Child_s500_v” and it will be spliced into “../data/ Child_s500_v1.csv”. The format of dataset should be “csv”, otherwise the program cannot execute.

The dataset is used for training. Columns are variables and rows are observations. Support both continuous and discrete sample datasets. There are some benchmark BN datasets in the data folder for using.

4th input = file number

The “file number” denotes the number of datasets used in one evaluation. For example, if file number=10, it means that the evaluation will run on 10 different datasets to evaluate the performance of certain MB algorithm. Moreover, The files must be of the same type and named sequentially, such as “../data/ Child_s500_v1.csv” to “../data/Child_s500_v9.csv”.

5th input = target variable index

If the number of nodes in a BN network is n, the index value of a node is among 0 to n-1. When users learn the MB sets of all nodes in the network, this parameter is set to "all". When users learn the MB sets of several nodes in the network, the input target node index need to be separated by ",". For example, "0,5,9" means an algorithm will return the MBs of nodes 0, 5, and 9.

6th input = alpha

The “alpha” denotes the significance level for conditional independence tests (e.g., χ^2 test and Fisher Z-test). The level of significance for hypothesis testing often is set 0.01 or 0.05.

7thinput = is _discrete

The “is_discrete” denotes whether an input dataset is discrete or continuous. We set “1” to denote a discrete dataset, and “0” to mean a continuous dataset.

Outputs (three outputs):

For local structure learning algorithms, the evaluation metrics include 1st output = reverse, 2nd output = miss, 3rd output = extra. The metric results will be written to the “*output*” folder.

The example of running MB-by-MB algorithm are shown in Figures 18 to 20 as follows:

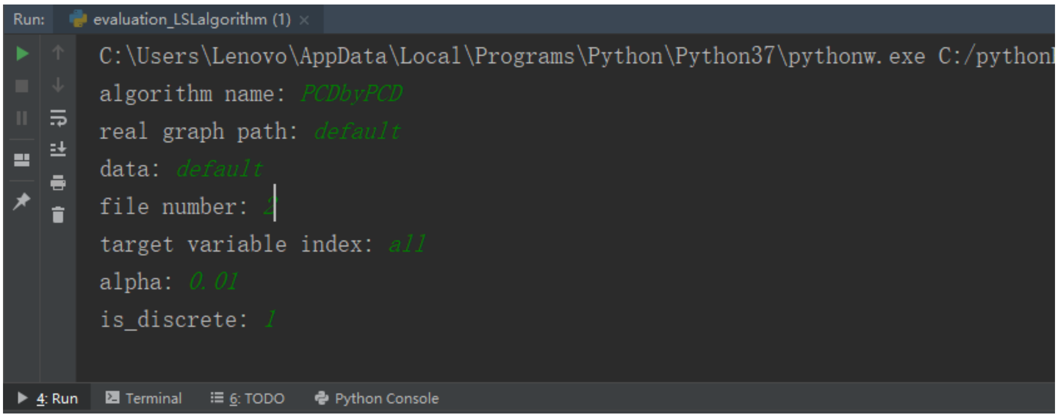


Figure 18 Before running the evaluation of PCD-by-PCD algorithm

Figure 19 The MBs of the target nodes after running PCD-by-PCD = algorithm

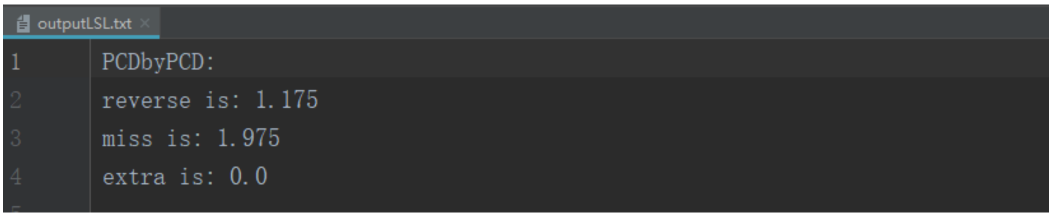


Figure 20 “outputLSL.txt” file after running PCD-by-PCD algorithm

Other example of the evaluation_LSL algorithms:

PCD-by-PCD algorithm:

algorithm name: PCDbyPCD real graph path: default data: default file number: 10 target variable index: all alpha: 0.01 is_discrete: 1
algorithm name: PCDbyPCD real graph path: ./data/child_graph.txt data: /data/Child_s500_v file number: 10 target variable index:0,1,5,9 alpha: 0.05 is_discrete: 0

MB-by-MB algorithm:

algorithm name: MBbyMB real graph path: default data: default file number: 10 target variable index: all alpha: 0.01 is_discrete: 1
algorithm name: MBbyMB real graph path: ./data/child_graph.txt data: ./data/Child_s500_v file number: 10 target variable index:0,1,5,9 alpha: 0.05 is_discrete: 0

CMB algorithm:

algorithm name: CMB real graph path: default data: default file number: 10 target variable index: all alpha: 0.01 is_discrete: 1
algorithm name: CMB real graph path: ./data/child_graph.txt data: ./data/Child_s500_v file number: 10 target variable index: 0,1,5,9 alpha: 0.05 is_discrete: 0

VI. MMHC algorithm

VI.1. Description

The MMHC algorithm is located at the "*pyBN*" folder. The MMHC algorithm was developed in the pyBN package. We integrated this algorithm into the pyCausalFS library. The "example_LSL" algorithm in the "*pyBN*" folder is used to learn an entire structure of a Bayesian network.

VI.2. Inputs and outputs of example_MMHC Algorithm

Table 13 Inputs and outputs of MMHC algorithm

Inputs (one parameter):	1st input = data
Outputs (one output):	1st output = structure

Inputs (one parameter):

1st input = data
The dataset is used for training. Columns are variables and rows are observations. Support both continuous and discrete sample datasets. There are some benchmark BN datasets in the data folder for using.

Outputs (one output):

1st output = structure
The output of structure learnt by MMHC will be shown in the terminal and written to the "*output_mmhc.txt*" file in the "*output*" folder.

The example of running MMHC algorithm are shown in Figures 21 to 23 as follows:

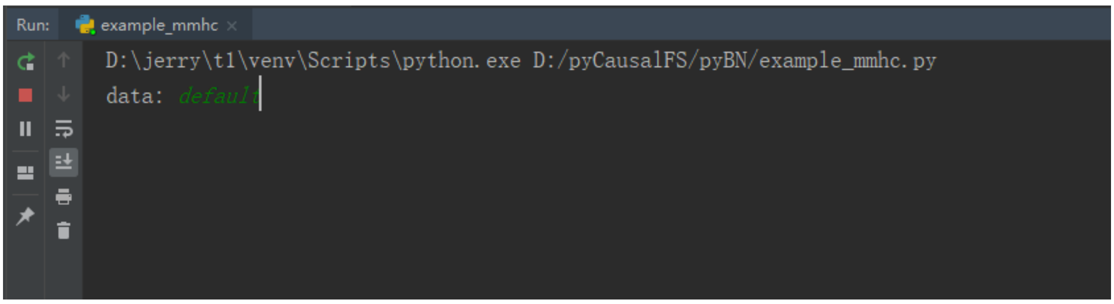


Figure 21 Before running the MMHC algorithm

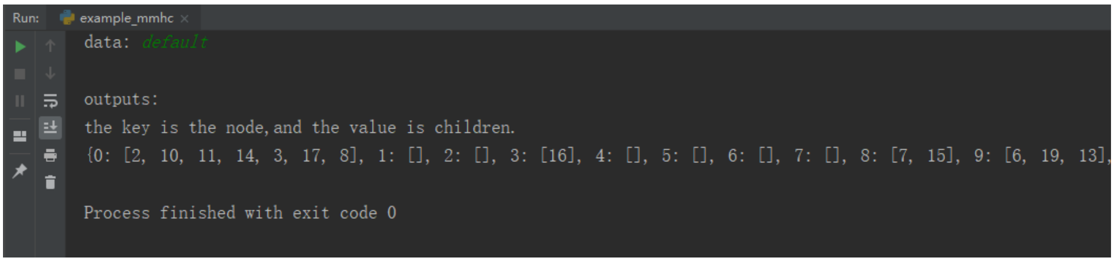


Figure 22 The structure of Bayesian network after running the MMHC algorithm

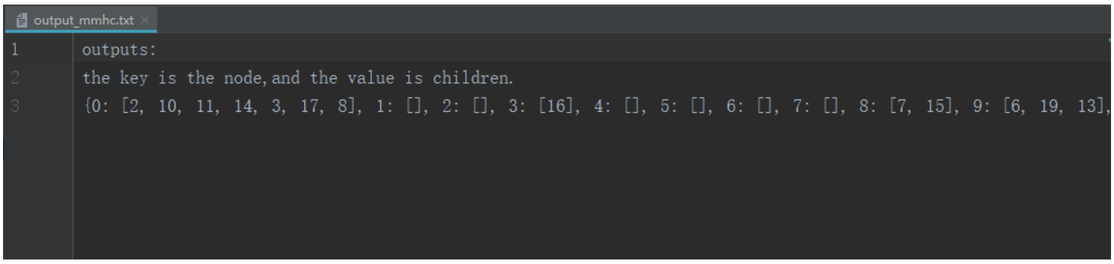


Figure 23 “*output_mmhc.txt*” file after running the MMHC algorithm