1. 首先订阅数据到自己的账户下

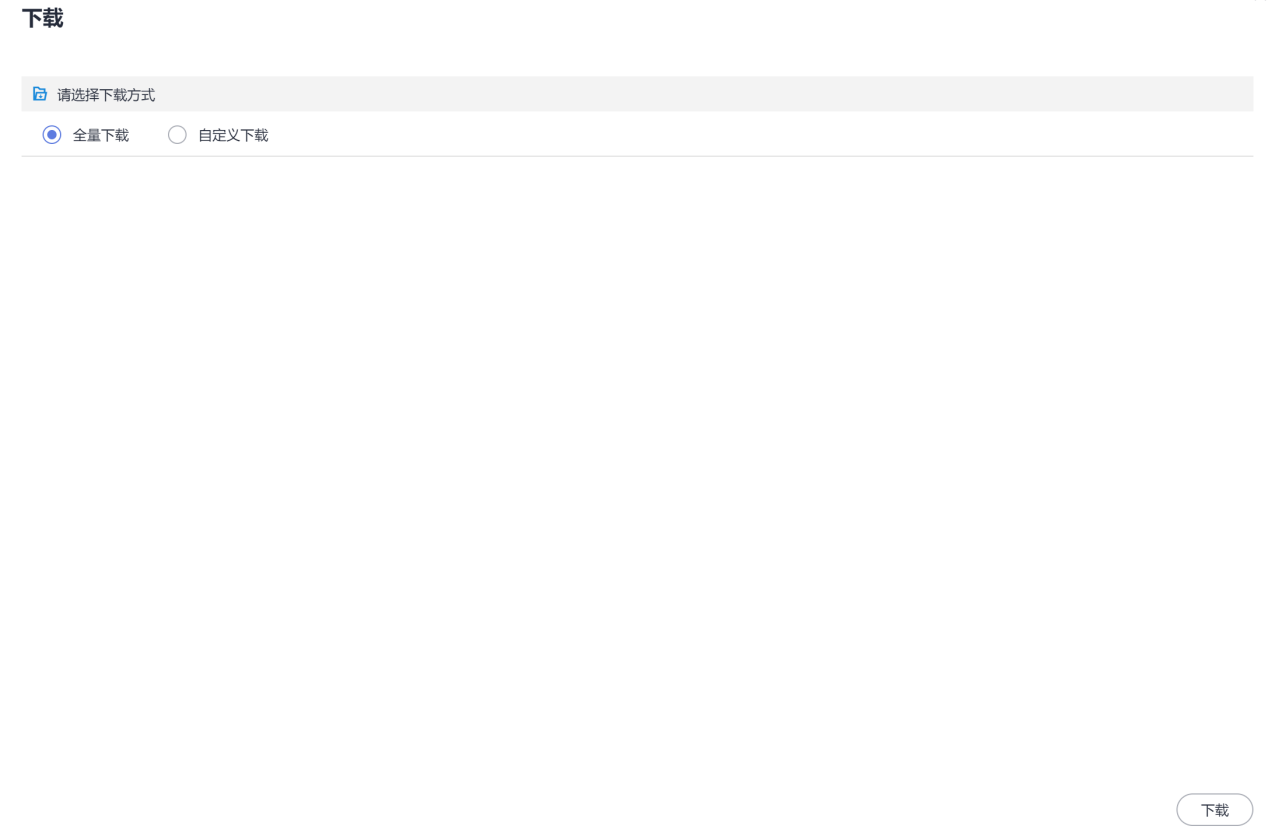




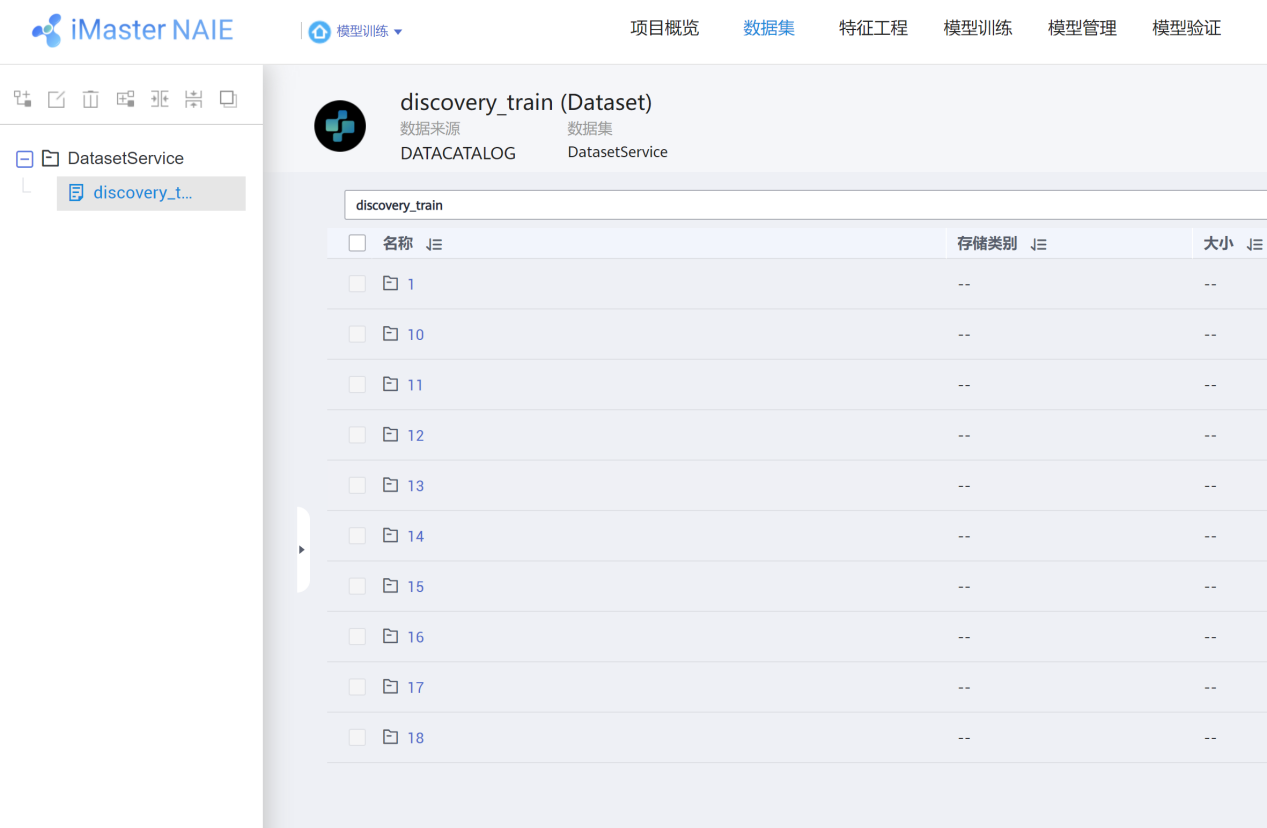




右边红框里的第二个蓝色小框，下载



全量下载，然后选择下载即可



可以回到自己的数据部分查看数据

1. 按照https://zhuanlan.zhihu.com/p/381957976 这里建立自己的训练环境（很类似，就不重复写了）
2. 如下修改requirements.txt和demo py文件，然后按照naie教程开始模型训练。（这里对于数据量很大的时候会比较耗时，尤其是最后两个数据集）
3. 其他参考文献：

[1] Zhou, H. Zha, and L. Song. Learning social infectivity in sparse low-rank networks using multi-dimensional hawkes processes. In AISTATS, 2013.

[2] Continuous time Bayesian network.

环境requirements.txt使用这个

#name [condition] [version]

#condition ==, >=, <=, >, <

#tensorflow==1.8.1

naie

tensorflow==1.15.2

gcastle==1.0.2

demo代码为：



## copy 如下

# -\*- coding: utf-8 -\*-

from \_\_future\_\_ import print\_function # do not delete this line if you want to save your log file.

import os

import pandas as pd

import numpy as np

from castle.common import GraphDAG

from castle.metrics import MetricsDAG

from castle.algorithms import TTPM

from castle.competition import submission

from naie.context import Context

import moxing as mox

from naie.datasets import get\_data\_reference

def arrs\_to\_csv(arrs, input\_path='submit.csv'):

"""

This can be used to generate the submission file in .csv

Parameters:

arrs: list of your solutions for each dataset; each element should be a numpy array of 0 or 1

input\_path: where to save your file; e.g., submit.csv

-------

"""

arrs\_str = [arr\_to\_string(arr) for arr in arrs]

pd.DataFrame(arrs\_str).to\_csv(input\_path, index=False)

# this copy the output for final submission

mox.file.copy(input\_path, os.path.join(Context.get\_output\_path(), 'submit.csv'))

def arr\_to\_string(mat):

"""

Parameters

mat: numpy array with each entry either 0 or 1

Returns:

string of the input array

"""

mat\_int = mat.astype(int)

mat\_flatten = mat\_int.flatten().tolist()

for m in mat\_flatten:

if m not in [0, 1]:

raise TypeError("Value not in {0, 1}.")

mat\_str = ' '.join(map(str, mat\_flatten))

return mat\_str

def remove\_diagnal\_entries(mat):

"""

set the diagonal of a matrix to be 0

"""

mat\_copy = np.copy(mat)

indices\_diag = np.diag\_indices(len(mat\_copy))

mat\_copy[indices\_diag] = 0

return mat\_copy

datasets = {}

## this is used for constructing dags with right shapes

mat\_shapes\_phase1 = [10, 11, 12, 13, 13, 14, 15, 16, 17, 18, 13, 20, 21, 16, 18, 24, 25, 26, 27, 29]

## import data; note that you need to first register dataset from naie

data\_reference = get\_data\_reference(dataset='DatasetService', dataset\_entity='discovery\_train', enable\_local\_cache=True)

for i in range(1, 21):

base\_dir = os.path.join('/cache/datasets/DatasetService/discovery\_train/', str(i))

X = pd.read\_csv(os.path.join(base\_dir, 'Alarm.csv'), encoding='utf-8')

## just to create a dag with right shapes, not the true dags

## we provided true dags for datasets 1-4, so you may use them here

true\_dag = np.ones((mat\_shapes\_phase1[i-1], mat\_shapes\_phase1[i-1]))

if os.path.exists(os.path.join(base\_dir, 'Topology.npy')):

topology\_matrix = np.load(os.path.join(base\_dir, 'Topology.npy'))

else:

topology\_matrix = None

datasets[i] = (X, topology\_matrix, true\_dag)

results = {}

for k in datasets:

print(k, datasets[k][0].shape)

X, topology\_matrix, true\_causal\_matrix = datasets[k]

train\_data = X.iloc[:, 0:3]

train\_data.columns = ['event', 'node', 'timestamp']

train\_data = train\_data.reindex(columns=['event', 'timestamp', 'node'])

if not isinstance(topology\_matrix, np.ndarray):

num\_nodes = len(set(train\_data['node']))

topology\_matrix = np.zeros((num\_nodes, num\_nodes))

ttpm = TTPM(topology\_matrix, max\_iter=20)

ttpm.learn(train\_data, max\_hop=0)

else:

ttpm = TTPM(topology\_matrix, max\_iter=20)

ttpm.learn(train\_data, max\_hop=1)

## with default setting, ttpm has diagonal entries, which means a node in the past may affect itself in the current

## the true graphs (DAGs) have ignored self-impact, so we remove the diagonal entries here

results[k] = (remove\_diagnal\_entries(ttpm.causal\_matrix.values), true\_causal\_matrix)

total\_rank\_score = 0

for k in range(1, 21):

est\_graph\_matrix, true\_causal\_matrix = results[k]

gscore = MetricsDAG(est\_graph\_matrix, true\_causal\_matrix).metrics['gscore']

total\_rank\_score += gscore

print(k)

print(MetricsDAG(est\_graph\_matrix, true\_causal\_matrix).metrics)

print('average score', total\_rank\_score/20)

### generate submission file

submit\_list = [results[k][0] for k in range(1,21)]

arrs\_to\_csv(submit\_list, input\_path='/cache/submit.csv')