dnn-inference

Ben Dai

Mar 21, 2023

CONTENTS

1 What We Can Do **3**

2 Reference **5**

3 Contents **7**

3.1 Installation 7

3.2 Examples 7

3.3 API Reference 15

4 Indices and tables **17**

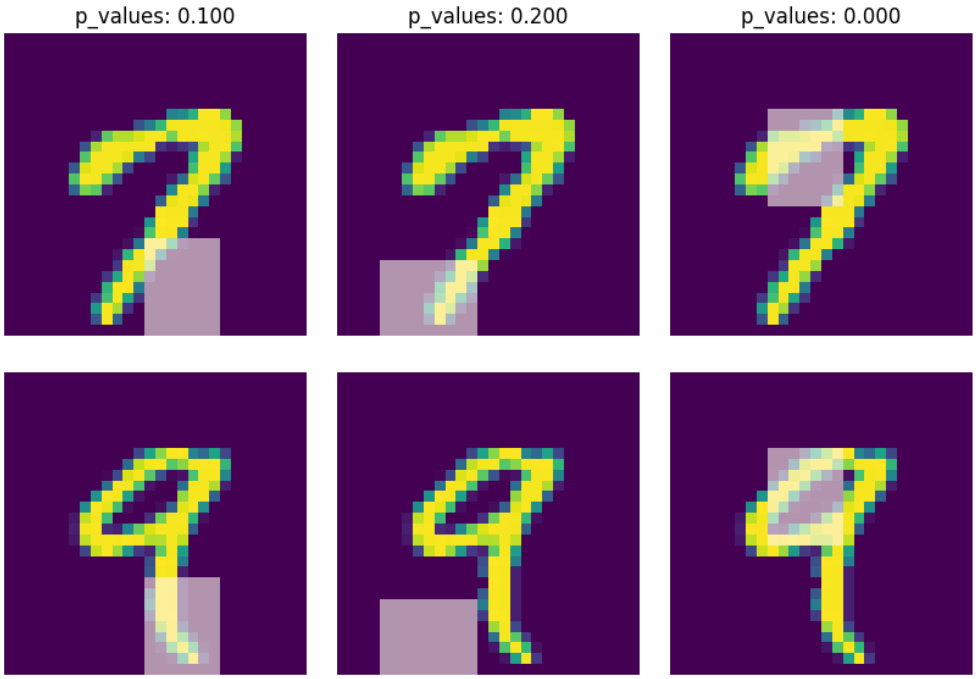
dnn-inference is a Python module for hypothesis testing based on blackbox models, including **deep neural networks**.

- GitHub repo: https://github.com/statmlben/dnn-inference
- Documentation: https://dnn-inference.readthedocs.io
- PyPi: https://pypi.org/project/dnn-inference
- Open Source: MIT license
- Paper: arXiv:2103.04985

CHAPTER

ONE

WHAT WE CAN DO



dnn-inference is able to provide an asymptotically valid *p-value* to examine if $\mathcal{S}$ is discriminative features to predict Y. Specifically, the proposed testing is:

$$H_0 : R(f^*) - R_{\mathcal{S}}(g^*) = 0, \quad \text{versus} \quad H_a : R(f^*) - R_{\mathcal{S}}(g^*) < 0,$$

where $\mathcal{S}$ is a collection of hypothesized features, R and $R_{\mathcal{S}}$ are risk functions with/without the hypothesized features $\mathbf{X}_{\mathcal{S}}$, and f^* and g^* are population minimizers on R and $R_{\mathcal{S}}$ respectively. The proposed test just considers the difference between the best predictive scores with/without hypothesized features. Please check more details in our paper arXiv:2103.04985.

- When *log-likelihood* is used as a loss function, then the test is equivalent to a conditional independence test: $Y \perp X_{\mathcal{S}} | X_{\mathcal{S}^c}$.
- Only *a small number of fitting* on neural networks is required, and the number can be as small as 1.
- Asymptotically Type I error control and power consistency.

CHAPTER

TWO

REFERENCE

If you use this code please star the repository and cite the following paper:

```
@article{dai2022significance,
title={Significance Tests of Feature Relevance for a Black-Box Learner},
author={Dai, Ben and Shen, Xiaotong and Pan, Wei},
journal={IEEE Transactions on Neural Networks and Learning Systems},
year={2022},
publisher={IEEE}
}
```

CHAPTER

THREE

CONTENTS

3.1 Installation

3.1.1 Dependencies

`dnn-inference` requires: **Python>=3.8** + [pip libs](./requirements.txt)

```
pip install -r requirements.txt
```

3.1.2 User installation

Install `dnn-inference` using `pip`

```
pip install dnn_inference
pip install git+https://github.com/statmlben/dnn-inference.git
```

3.1.3 Source code

You can check the latest sources with the command.

```
git clone https://github.com/statmlben/dnn-inference.git
```

3.2 Examples

Contents

- *Examples*
  - *MNIST dataset*
  - *Boston Housing Dataset*

3.2.1 MNIST dataset

`.sig_test.split_test` in MNIST dataset

[2]:
```
import numpy as np
from tensorflow import keras
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPooling2D
from tensorflow.python.keras import backend as K
import time
from sklearn.model_selection import train_test_split
from tensorflow.keras.optimizers import Adam, SGD
```

[4]:
```
np.random.seed(0)
num_classes = 2
# input image dimensions
img_rows, img_cols = 28, 28

# the data, split between train and test sets
(x_train, y_train), (x_test, y_test) = mnist.load_data()
X = np.vstack((x_train, x_test))
y = np.hstack((y_train, y_test))
ind = (y == 9) + (y == 7)
X, y = X[ind], y[ind]
X = X.astype('float32')
X += .01*abs(np.random.randn(14251, 28, 28))
y[y==7], y[y==9] = 0, 1

if K.image_data_format() == 'channels_first':
    X = X.reshape(x.shape[0], 1, img_rows, img_cols)
    input_shape = (1, img_rows, img_cols)
else:
    X = X.reshape(X.shape[0], img_rows, img_cols, 1)
    input_shape = (img_rows, img_cols, 1)

X /= 255.

# convert class vectors to binary class matrices
y = keras.utils.to_categorical(y, num_classes)
```

[5]:
```
## define the learning models
def cnn():
    model = Sequential()
    model.add(Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=input_shape))
    model.add(Conv2D(64, (3, 3), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.25))
    model.add(Flatten())
    model.add(Dense(128, activation='relu'))
    model.add(Dropout(0.5))
    model.add(Dense(num_classes, activation='softmax'))
```

(continues on next page)

(continued from previous page)

```
    model.compile(loss=keras.losses.binary_crossentropy, optimizer=keras.optimizers.
↪Adam(0.0005), metrics=['accuracy'])
    return model

model_null, model_alter = cnn(), cnn()
```

```
2022-06-29 16:58:55.239237: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:936]␣
↪successful NUMA node read from SysFS had negative value (-1), but there must be at␣
↪least one NUMA node, so returning NUMA node zero
2022-06-29 16:58:55.243474: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:936]␣
↪successful NUMA node read from SysFS had negative value (-1), but there must be at␣
↪least one NUMA node, so returning NUMA node zero
2022-06-29 16:58:55.243767: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:936]␣
↪successful NUMA node read from SysFS had negative value (-1), but there must be at␣
↪least one NUMA node, so returning NUMA node zero
2022-06-29 16:58:55.244460: I tensorflow/core/platform/cpu_feature_guard.cc:151] This␣
↪TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to use␣
↪the following CPU instructions in performance-critical operations:  AVX2 AVX512F FMA
To enable them in other operations, rebuild TensorFlow with the appropriate compiler␣
↪flags.
2022-06-29 16:58:55.245170: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:936]␣
↪successful NUMA node read from SysFS had negative value (-1), but there must be at␣
↪least one NUMA node, so returning NUMA node zero
2022-06-29 16:58:55.245475: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:936]␣
↪successful NUMA node read from SysFS had negative value (-1), but there must be at␣
↪least one NUMA node, so returning NUMA node zero
2022-06-29 16:58:55.245750: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:936]␣
↪successful NUMA node read from SysFS had negative value (-1), but there must be at␣
↪least one NUMA node, so returning NUMA node zero
2022-06-29 16:58:55.623115: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:936]␣
↪successful NUMA node read from SysFS had negative value (-1), but there must be at␣
↪least one NUMA node, so returning NUMA node zero
2022-06-29 16:58:55.623425: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:936]␣
↪successful NUMA node read from SysFS had negative value (-1), but there must be at␣
↪least one NUMA node, so returning NUMA node zero
2022-06-29 16:58:55.623674: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:936]␣
↪successful NUMA node read from SysFS had negative value (-1), but there must be at␣
↪least one NUMA node, so returning NUMA node zero
2022-06-29 16:58:55.623925: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1525]␣
↪Created device /job:localhost/replica:0/task:0/device:GPU:0 with 3763 MB memory:  ->␣
↪device: 0, name: NVIDIA GeForce RTX 2060, pci bus id: 0000:01:00.0, compute capability:
↪ 7.5
```

[6]:
```
## fitting param
from tensorflow.keras.callbacks import EarlyStopping
es = EarlyStopping(monitor='val_accuracy', mode='max', verbose=0, patience=15, restore_
↪best_weights=True)

fit_params = {'callbacks': [es],
              'epochs': 50,
              'batch_size': 32,
              'validation_split': .2,
```

(continues on next page)

(continued from previous page)

```
            'verbose': 0}

## testing params
test_params = { 'split': "one-split",
                'inf_ratio': None,
                'perturb': None,
                'cv_num': 2,
                'cp': 'hommel',
                'verbose': 2}

## tuning params
tune_params = { 'num_perm': 100,
                'ratio_grid': [.2, .4, .6, .8],
                'if_reverse': 0,
                'perturb_range': 2.**np.arange(-3,3,.1),
                'tune_ratio_method': 'fuse',
                'tune_pb_method': 'fuse',
                'cv_num': 2,
                'cp': 'hommel',
                'verbose': 2}
```

[7]:
```
## Inference based on dnn_inference
from dnn_inference.sig_test import split_test
## testing based on learning models
inf_feats = [[np.arange(19,28), np.arange(13,20)], [np.arange(21,28), np.arange(4, 13)],
↪[np.arange(7,16), np.arange(9,16)]]
cue = split_test(inf_feats=inf_feats, model_null=model_null, model_alter=model_alter,␣
↪eva_metric='zero-one')
P_value = cue.testing(X, y, fit_params, test_params, tune_params)
```

```
INFO:tensorflow:Assets written to: ./saved/split_test/06-29_16-59/model_null_init/assets
```

```
2022-06-29 16:59:00.569380: W tensorflow/python/util/util.cc:368] Sets are not currently␣
↪considered sequences, but this may change in the future, so consider avoiding using␣
↪them.
```

```
INFO:tensorflow:Assets written to: ./saved/split_test/06-29_16-59/model_alter_init/assets
====================== one-split for 0-th Hypothesis ======================
```

```
2022-06-29 16:59:02.617064: I tensorflow/stream_executor/cuda/cuda_dnn.cc:368] Loaded␣
↪cuDNN version 8101
```

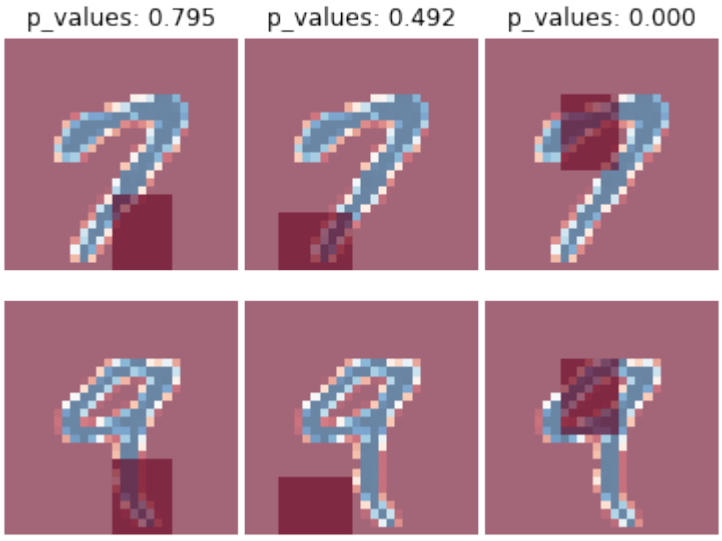
```
(tuneHP: ratio) Est. Type 1 error: 0.000; inf sample ratio: 0.200
 (tuneHP: ratio) Done with inf sample ratio: 0.200
(tuneHP: pb) Est. Type 1 error: 0.020; perturbation level: 0.125
 (tuneHP: pb) Done with inf pb level: 0.125
cv: 0; p_value: 0.26510; loss_null: 0.00140(0.03744); loss_alter: 0.00175(0.04185)
cv: 1; p_value: 0.68497; loss_null: 0.00211(0.04583); loss_alter: 0.00175(0.04185)
  0-th Hypothesis: accept H0 with p_value: 0.795
====================== one-split for 1-th Hypothesis ======================
(tuneHP: ratio) Est. Type 1 error: 0.000; inf sample ratio: 0.200
 (tuneHP: ratio) Done with inf sample ratio: 0.200
(tuneHP: pb) Est. Type 1 error: 0.000; perturbation level: 0.125
 (tuneHP: pb) Done with inf pb level: 0.125
```

(continues on next page)

(continued from previous page)

```
cv: 0; p_value: 0.16398; loss_null: 0.00140(0.03744); loss_alter: 0.00211(0.04583)
cv: 1; p_value: 0.92602; loss_null: 0.00316(0.05611); loss_alter: 0.00175(0.04185)
  1-th Hypothesis: accept H0 with p_value: 0.492
====================== one-split for 2-th Hypothesis ======================
(tuneHP: ratio) Est. Type 1 error: 0.020; inf sample ratio: 0.200
 (tuneHP: ratio) Done with inf sample ratio: 0.200
(tuneHP: pb) Est. Type 1 error: 0.100; perturbation level: 0.125
(tuneHP: pb) Est. Type 1 error: 0.110; perturbation level: 0.134
(tuneHP: pb) Est. Type 1 error: 0.130; perturbation level: 0.144
(tuneHP: pb) Est. Type 1 error: 0.100; perturbation level: 0.154
(tuneHP: pb) Est. Type 1 error: 0.090; perturbation level: 0.165
(tuneHP: pb) Est. Type 1 error: 0.120; perturbation level: 0.177
(tuneHP: pb) Est. Type 1 error: 0.120; perturbation level: 0.189
(tuneHP: pb) Est. Type 1 error: 0.120; perturbation level: 0.203
(tuneHP: pb) Est. Type 1 error: 0.110; perturbation level: 0.218
(tuneHP: pb) Est. Type 1 error: 0.110; perturbation level: 0.233
(tuneHP: pb) Est. Type 1 error: 0.150; perturbation level: 0.250
(tuneHP: pb) Est. Type 1 error: 0.130; perturbation level: 0.268
(tuneHP: pb) Est. Type 1 error: 0.100; perturbation level: 0.287
(tuneHP: pb) Est. Type 1 error: 0.120; perturbation level: 0.308
(tuneHP: pb) Est. Type 1 error: 0.090; perturbation level: 0.330
(tuneHP: pb) Est. Type 1 error: 0.060; perturbation level: 0.354
(tuneHP: pb) Est. Type 1 error: 0.070; perturbation level: 0.379
(tuneHP: pb) Est. Type 1 error: 0.040; perturbation level: 0.406
 (tuneHP: pb) Done with inf pb level: 0.406
cv: 0; p_value: 0.00000; loss_null: 0.00246(0.04950); loss_alter: 0.03088(0.17298)
cv: 1; p_value: 0.00000; loss_null: 0.00175(0.04185); loss_alter: 0.03298(0.17859)
  2-th Hypothesis: reject H0 with p_value: 0.000
```

[8]:
```
## visualize testing results
cue.visual(X,y)
print('P-values: %s' %P_value)
```



```
P-values: [0.7953140193691346, 0.49192858264606976, 1.1876805112986783e-19]
```

3.2.2 Boston Housing Dataset

.sig_test.split_test in Boston Housing price regression dataset

[1]:
```
import numpy as np
import tensorflow.keras as keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPooling2D
from tensorflow.python.keras import backend as K
import time
from sklearn.model_selection import train_test_split
from tensorflow.keras.optimizers import Adam, SGD
from dnn_inference.sig_test import split_test
import tensorflow as tf
```

Boston house prices dataset

Data Set Characteristics:

```
Number of Instances: 506

Number of Attributes: 13 numeric/categorical predictive. Median Value (attribute 14) is␣
↪usually the target.

Attribute Information (in order)
    - CRIM     per capita crime rate by town
    - ZN       proportion of residential land zoned for lots over 25,000 sq.ft.
    - INDUS    proportion of non-retail business acres per town
    - CHAS     Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)
    - NOX      nitric oxides concentration (parts per 10 million)
    - RM       average number of rooms per dwelling
    - AGE      proportion of owner-occupied units built prior to 1940
    - DIS      weighted distances to five Boston employment centres
    - RAD      index of accessibility to radial highways
    - TAX      full-value property-tax rate per USD10,000
    - PTRATIO  pupil-teacher ratio by town
    - B        1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town
    - LSTAT    % lower status of the population
    - MEDV     Median value of owner-occupied homes in $1000's
```

[2]:
```
(x_train, y_train), (_, _) = tf.keras.datasets.boston_housing.load_data(path="boston_
↪housing.npz",
                                                                        test_
↪split=0.1)
y_train, y_test = y_train[:,np.newaxis], y_test[:,np.newaxis]

from sklearn import preprocessing
scaler = preprocessing.MinMaxScaler()
x_train = scaler.fit_transform(x_train)
```

[3]:
```
n, d = x_train.shape
print('num of samples: %d, dim: %d' %(n, d))
```

```
num of samples: 455, dim: 13
```

[4]:
```
from tensorflow import keras
from tensorflow.keras import layers

def build_model():
  model = keras.Sequential([
    layers.Dense(8, activation='relu', input_shape=[d]),
    layers.BatchNormalization(),
    layers.Dense(1, activation='relu')
  ])

  optimizer = tf.keras.optimizers.Adam(1e-3)

  model.compile(loss='mse',
                optimizer=optimizer,
                metrics=['mae', 'mse'])
  return model
```

[5]:
```
model_null, model_alter = build_model(), build_model()
```

```
2022-06-30 12:53:52.290176: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:936]␣
↪successful NUMA node read from SysFS had negative value (-1), but there must be at␣
↪least one NUMA node, so returning NUMA node zero
2022-06-30 12:53:52.295414: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:936]␣
↪successful NUMA node read from SysFS had negative value (-1), but there must be at␣
↪least one NUMA node, so returning NUMA node zero
2022-06-30 12:53:52.295682: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:936]␣
↪successful NUMA node read from SysFS had negative value (-1), but there must be at␣
↪least one NUMA node, so returning NUMA node zero
2022-06-30 12:53:52.296167: I tensorflow/core/platform/cpu_feature_guard.cc:151] This␣
↪TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to use␣
↪the following CPU instructions in performance-critical operations:  AVX2 AVX512F FMA
To enable them in other operations, rebuild TensorFlow with the appropriate compiler␣
↪flags.
2022-06-30 12:53:52.296992: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:936]␣
↪successful NUMA node read from SysFS had negative value (-1), but there must be at␣
↪least one NUMA node, so returning NUMA node zero
2022-06-30 12:53:52.297257: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:936]␣
↪successful NUMA node read from SysFS had negative value (-1), but there must be at␣
↪least one NUMA node, so returning NUMA node zero
2022-06-30 12:53:52.297500: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:936]␣
↪successful NUMA node read from SysFS had negative value (-1), but there must be at␣
↪least one NUMA node, so returning NUMA node zero
2022-06-30 12:53:52.641227: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:936]␣
↪successful NUMA node read from SysFS had negative value (-1), but there must be at␣
↪least one NUMA node, so returning NUMA node zero
2022-06-30 12:53:52.641520: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:936]␣
↪successful NUMA node read from SysFS had negative value (-1), but there must be at␣
↪least one NUMA node, so returning NUMA node zero
```

(continues on next page)

(continued from previous page)

```
2022-06-30 12:53:52.641753: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:936]␣
↪successful NUMA node read from SysFS had negative value (-1), but there must be at␣
↪least one NUMA node, so returning NUMA node zero
2022-06-30 12:53:52.642007: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1525]␣
↪Created device /job:localhost/replica:0/task:0/device:GPU:0 with 3022 MB memory:  ->␣
↪device: 0, name: NVIDIA GeForce RTX 2060, pci bus id: 0000:01:00.0, compute capability:
↪ 7.5
```

[6]:
```
from tensorflow.keras.callbacks import EarlyStopping
es = EarlyStopping(monitor='val_loss', mode='min',
                                   verbose=0, patience=300, restore_best_weights=True)

fit_params = {'callbacks': [es],
                      'epochs': 3000,
                      'batch_size': 32,
                      'validation_split': .2,
                      'verbose': 0}

## testing params
test_params = { 'split': "one-split",
                'inf_ratio': None,
                'perturb': None,
                'cv_num': 2,
                'cp': 'hommel',
                'verbose': 2}

## tuning params
tune_params = { 'num_perm': 100,
                'ratio_grid': [.2, .4, .6, .8],
                'if_reverse': 1,
                'perturb_range': 2.**np.arange(-3,3,.3),
                'tune_ratio_method': 'fuse',
                'tune_pb_method': 'fuse',
                'cv_num': 2,
                'cp': 'hommel',
                'verbose': 2}
```

[7]:
```
inf_feats = [np.arange(3), np.arange(5,11)]

cue = split_test(inf_feats=inf_feats, model_null=model_null, model_alter=model_alter,␣
↪eva_metric='mse')

P_value = cue.testing(x_train, y_train, fit_params, test_params, tune_params)
```

```
INFO:tensorflow:Assets written to: ./saved/split_test/06-30_12-53/model_null_init/assets
```

```
2022-06-30 12:53:59.491087: W tensorflow/python/util/util.cc:368] Sets are not currently␣
↪considered sequences, but this may change in the future, so consider avoiding using␣
↪them.
```

```
INFO:tensorflow:Assets written to: ./saved/split_test/06-30_12-53/model_alter_init/assets
==================== one-split test for 0-th Hypothesis ====================
```

(continues on next page)

(continued from previous page)

```
(tuneHP: ratio) Est. Type 1 error: 0.150; inf sample ratio: 0.800
(tuneHP: ratio) Est. Type 1 error: 0.270; inf sample ratio: 0.600
(tuneHP: ratio) Est. Type 1 error: 0.000; inf sample ratio: 0.400
 (tuneHP: ratio) Done with inf sample ratio: 0.400
(tuneHP: pb) Est. Type 1 error: 0.000; perturbation level: 0.125
 (tuneHP: pb) Done with inf pb level: 0.125
cv: 0; p_value: 0.03247; loss_null: 14.63722(51.76234); loss_alter: 18.41512(42.01610)
cv: 1; p_value: 0.01776; loss_null: 14.65350(46.57168); loss_alter: 21.16354(51.60448)
  0-th Hypothesis: reject H0 with p_value: 0.049
==================== one-split test for 1-th Hypothesis ====================
(tuneHP: ratio) Est. Type 1 error: 1.000; inf sample ratio: 0.800
(tuneHP: ratio) Est. Type 1 error: 0.000; inf sample ratio: 0.600
 (tuneHP: ratio) Done with inf sample ratio: 0.600
(tuneHP: pb) Est. Type 1 error: 0.700; perturbation level: 0.125
(tuneHP: pb) Est. Type 1 error: 0.730; perturbation level: 0.154
(tuneHP: pb) Est. Type 1 error: 0.700; perturbation level: 0.189
(tuneHP: pb) Est. Type 1 error: 0.700; perturbation level: 0.233
(tuneHP: pb) Est. Type 1 error: 0.610; perturbation level: 0.287
(tuneHP: pb) Est. Type 1 error: 0.580; perturbation level: 0.354
(tuneHP: pb) Est. Type 1 error: 0.470; perturbation level: 0.435
(tuneHP: pb) Est. Type 1 error: 0.460; perturbation level: 0.536
(tuneHP: pb) Est. Type 1 error: 0.390; perturbation level: 0.660
(tuneHP: pb) Est. Type 1 error: 0.290; perturbation level: 0.812
(tuneHP: pb) Est. Type 1 error: 0.180; perturbation level: 1.000
(tuneHP: pb) Est. Type 1 error: 0.190; perturbation level: 1.231
(tuneHP: pb) Est. Type 1 error: 0.170; perturbation level: 1.516
(tuneHP: pb) Est. Type 1 error: 0.070; perturbation level: 1.866
(tuneHP: pb) Est. Type 1 error: 0.070; perturbation level: 2.297
(tuneHP: pb) Est. Type 1 error: 0.030; perturbation level: 2.828
 (tuneHP: pb) Done with inf pb level: 2.828
cv: 0; p_value: 0.12088; loss_null: 31.87867(120.14606); loss_alter: 34.30483(86.49743)
cv: 1; p_value: 0.83584; loss_null: 33.45124(124.96815); loss_alter: 33.72868(86.60861)
  1-th Hypothesis: accept H0 with p_value: 0.363
```

[9]:
```
P_value
```

[9]:
```
[0.048711537430423474, 0.3626513098329711]
```

3.3 API Reference

Contents

- *API Reference*
  - *dnn_inference.sig_test*
    - *split_test*
    - *perm_test*
    - *Hperm_test*

3.3.1 dnn_inference.sig_test

split_test

perm_test

Hperm_test

CHAPTER

FOUR

INDICES AND TABLES

- genindex
- search