anko

0.2.1

Generated by Doxygen 1.8.17

Chapter 1

anko

Toolkit for performing anomaly detection algorithm on 1D time series based on numpy, scipy.

1.1 Requirements:

- numpy >= 1.16.4
- scipy >= 1.2.1

1.2 Installation:

```
pip install anko
```

1.3 Documentation:

anko

1.4 Jupyter Notebook Tutorial (in dev):

host on mybinder

1.5 Basic Usage:

- First step: Call AnomalyDetector
  ```
  from anko.anomaly_detector import AnomalyDetector
  agent = AnomalyDetector(t, series)
  ```
- Second step: Define policies and threshold values (optional)
  ```
  agent.thres_params["linregress_res"] = 1.5
  agent.apply_policies["z_normalization"] = True
  agent.apply_policies["info_criterion"] = 'AIC'
  ```
- Third step: Run check
  ```
  check_result = agent.check()
  ```
  The type of output **check_result** is **anko.anomaly_detector.CheckResult**, which is basically a dictionary.
  ```
  model: 'increase_step_func'
  popt: [220.3243250055105, 249.03846355234577, 74.00000107457113]
  perr: [0.4247789247961187, 0.7166253174634686, 0.0]
  anomalous_data: [(59, 209)]
  residual: [10.050378152592119]
  extra_info: ['Info: AnomalyDetector is using z normalization.', 'Info: There are more than 1
   discontinuous points detected.']
  ```

1.6 Run Test (in dev):

```
python -m unittest discover -s test -p '*_test.py'
```

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 4

Class Index

4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

Chapter 6

Namespace Documentation

6.1 anko Namespace Reference

Namespaces

- anomaly_detector
- stats_util

6.2 anko.anomaly_detector Namespace Reference

Classes

- class AnomalousData
- class AnomalyDetector
- class CheckResult

6.3 anko.stats_util Namespace Reference

Functions

- def get_histogram (np.ndarray x, bool sort_histo=False)

  Return the corresponding histogram of the data x.
- np.ndarray normal_distr (np.ndarray x, float a, float x0, float sigma)

  Calculate normal distribution of input array x.
- def gaussian_fit (np.ndarray x, bool sort_histo=False, str half=None, int maxfev=2000, bounds=[0, 1e+6])

  Fitting the Gaussian (normal) distribution for input data x.
- np.ndarray left_half_normal_distr (np.ndarray x, float a, float x0, float sigma)

  Calculate left-side half normal distribution of input array x.
- np.ndarray right_half_normal_distr (np.ndarray x, float a, float x0, float sigma)

  Calculate right-side half normal distribution of input array x.
- def flat_histogram (np.ndarray x)

  Manually assign parameters of Gaussian distrinution if the given histogram is too flat.
- def linear_regression (np.ndarray x, np.ndarray y)

Fitting linear ansatz for input data (x, y).

- bool data_is_linear (np.ndarray x, np.ndarray y, float std_err_th=1e-2)

  Check whether the data (x, y) is linear under the given tolerance.
- np.ndarray general_erf (np.ndarray x, float a, float b, float x0)

  Calculate the generalize error function of input array x.
- np.ndarray three_stair_erf (np.ndarray x, float c0, float c1, float c2, float x1, float x2)
- def general_erf_fit (np.ndarray x, np.ndarray y, bool three_stair=False, int maxfev=2000, bounds=[0, 1e+6])

  Fitting generalize error function for input data (x, y).
- np.ndarray exp_decay (np.ndarray x, float a, float alpha)

  Calculate the exponential function of input array x.
- def exp_decay_fit (np.ndarray x, np.ndarray y, str mode='log-linregress', int maxfev=2000, bounds=[-1e-6, 1e+6])
- def smoothness (np.ndarray x, bool normalize=False)
- def discontinuous_idx (np.ndarray x, int std_width=1)
- bool is_oscillating (np.ndarray x, float osci_freq_th=0.3)

  Determine whether the input array x is oscillating over its mean with frequency larger than osci_freq_th.
- np.ndarray fitting_residual (np.ndarray x, np.ndarray y, func, args, float mask_min=None, bool absolute_↩ value=True, bool standardized=False)

  Compute the fitting residual.
- float AIC_score (np.ndarray y, np.ndarray y_predict, int p)

  Compute Akaike information criterion for model selection.
- float BIC_score (np.ndarray y, np.ndarray y_predict, int p)

  Compute Bayesian information criterion for model selection.
- np.ndarray z_normalization (np.ndarray x)

  Perform z-score normalizaion on input array x.

6.3.1 Function Documentation

6.3.1.1 AIC_score()

```
float anko.stats_util.AIC_score (
            np.ndarray y,
            np.ndarray y_predict,
            int p )
```

Compute Akaike information criterion for model selection.

Parameters

y	(numpy.ndarray): Data samples.
y_predict	(numpy.ndarray): Prediction by fitting.
p	(int): Fitting degrees of freedom, i.e. the number of parameters to fit with.

Returns

aic_score (float):

6.3.1.2 BIC_score()

```
float anko.stats_util.BIC_score (
            np.ndarray y,
            np.ndarray y_predict,
            int p )
```

Compute Bayesian information criterion for model selection.

Parameters

y	(numpy.ndarray): Data samples.
y_predict	(numpy.ndarray): Prediction by fitting.
p	(int): Fitting degrees of freedom, i.e. the number of parameters to fit with.

Returns

bic_score (float):

6.3.1.3 data_is_linear()

```
bool anko.stats_util.data_is_linear (
            np.ndarray x,
            np.ndarray y,
            float std_err_th = 1e-2 )
```

Check whether the data (x, y) is linear under the given tolerance.

This will perform a linear regression fitting.

Parameters

x	(numpy.ndarray): x coordinate of input data points.
y	(numpy.ndarray): y coordinate of input data points.
std_err_th	(float, optional): Threshold value of std_err.

Returns

out (bool): Return Ture if data is flat, else return False.

6.3.1.4 discontinuous_idx()

```
def anko.stats_util.discontinuous_idx (
            np.ndarray x,
            int std_width = 1 )
```

Parameters

x	(numpy.↩ ndarray):
std_width	(int):

Returns

idx (numpy.ndarray):

6.3.1.5 exp_decay()

```
np.ndarray anko.stats_util.exp_decay (
              np.ndarray x,
              float a,
              float alpha )
```

Calculate the exponential function of input array x.

Note that Domain of x >= 0.

Parameters

x	(numpy.↩ ndarray):
a	(float):
alpha	(float):

Returns

out:

6.3.1.6 exp_decay_fit()

```
def anko.stats_util.exp_decay_fit (
              np.ndarray x,
              np.ndarray y,
              str mode = 'log-linregress',
              int maxfev = 2000,
               bounds = [-1e-6,1e+6] )
```

Parameters

x	(numpy.↩ ndarray):

Parameters

y	(numpy.↩ ndarray):
mode	(str):
maxfev	(int):
bounds	(list[float,float])↩ :

Returns

popt (numpy.ndarray):

perr (numpy.ndarray):

6.3.1.7 fitting_residual()

```
np.ndarray anko.stats_util.fitting_residual (
             np.ndarray x,
             np.ndarray y,
              func,
              args,
             float mask_min = None,
             bool absolute_value = True,
             bool standardized = False )
```

Compute the fitting residual.

Parameters

x	(numpy.ndarray): x coordinate of input data points.
y	(numpy.ndarray): y coordinate of input data points.
func	(callable): Fitting function.
args	(numpy.ndarray): Best estimated arguments of fitting function.
mask_min	(float, optional): If not None, mask resuduals that are smaller than mask_min to zero. This is always performed before standardization.
absolute_value	(bool, optional): If True, return absolute value of residual.
standardized	(bool, optional): Standardize residual to z-score formalism.

Returns

res (numpy.ndarray): Residual of each corresponding data points (x, y).

6.3.1.8 flat_histogram()

```
def anko.stats_util.flat_histogram (
             np.ndarray x )
```

Manually assign parameters of Gaussian distrinution if the given histogram is too flat.

In this senario the histogram of data is regarded as a local segment of a larger normal-distribution-like histogram, with standard deviation which exceeds the current consideration of domain.

Parameters of Gaussian distribution are assigned as following:

1. Number of appearance of mode as normalization constant, a.
2. Mode of data x as mean, x0.
3. Standard deviation is set to infinity (numpy.inf).

Parameters

x	(numpy.ndarray): Input values.

Returns

popt (numpy.ndarray): Assigned values for Gaussian distribution.

perr (numpy.ndarray): Errors are set to zero.

6.3.1.9 gaussian_fit()

```
def anko.stats_util.gaussian_fit (
             np.ndarray x,
             bool sort_histo = False,
             str half = None,
             int maxfev = 2000,
              bounds = [0,1e+6] )
```

Fitting the Gaussian (normal) distribution for input data x.

Parameters

x	(numpy.ndarray): Input values.
sort_histo	(bool, optional): If True use the sorted histogram.
maxfev	(int, optional): Maximum step of fitting iteration.

Returns

popt (numpy.ndarray): Estimate value of a, x0 and sigma of Gaussian distribution.

perr (numpy.ndarray): Error of popt. Defined by the square of diagonal element of covariance matrix.

6.3.1.10 general_erf()

```
np.ndarray anko.stats_util.general_erf (
              np.ndarray x,
              float a,
              float b,
              float x0 )
```

Calculate the generalize error function of input array x.

f(x) = a; x $<$ x0, (a+b)/2; x = x0, b; x $>$ x0

Parameters

x	(numpy.ndarray): Input values.
a	(float): Value of first stair.
b	(float): Value of second stair.
x0	(float): Location of the cliff.

Returns

out (numpy.ndarray): Output array.

6.3.1.11 general_erf_fit()

```
def anko.stats_util.general_erf_fit (
              np.ndarray x,
              np.ndarray y,
              bool three_stair = False,
              int maxfev = 2000,
               bounds = [0,1e+6] )
```

Fitting generalize error function for input data (x, y).

Parameters

x	(numpy.↩ ndarray):
y	(numpy.↩ ndarray):
three_stair	(bool):
maxfev	(int):
bounds	(list[float]):

Returns

popt (numpy.ndarray):

perr (numpy.ndarray):

6.3.1.12 get_histogram()

```
def anko.stats_util.get_histogram (
            np.ndarray x,
            bool sort_histo = False )
```

Return the corresponding histogram of the data x.

Parameters

x	(numpy.ndarray): One-dimensional array of data.
sort_histo	(bool, optional): If True return the sorted histogram.

Returns

keys: Set of data x (no duplicate).

vals: Number of appearance for each key in keys.

6.3.1.13 is_oscillating()

```
bool anko.stats_util.is_oscillating (
            np.ndarray x,
            float osci_freq_th = 0.3 )
```

Determine whether the input array x is oscillating over its mean with frequency larger than osci_freq_th.

Parameters

x	(numpy.↩ ndarray):
osci_freq↩ _th	(float):

Returns

out (bool):

6.3.1.14 left_half_normal_distr()

```
np.ndarray anko.stats_util.left_half_normal_distr (
            np.ndarray x,
            float a,
            float x0,
            float sigma )
```

Calculate left-side half normal distribution of input array x.

Parameters

x	(numpy.ndarray): Input values.
a	(float): Overall normalization constant.
x0	(float): Mean.
sigma	(float): Standard deviation.

Returns

out (numpy.ndarray): Output array.

6.3.1.15 linear_regression()

```
def anko.stats_util.linear_regression (
            np.ndarray x,
            np.ndarray y )
```

Fitting linear ansatz for input data (x, y).

y = intercept + slope ∗ x.

Parameters

x	(numpy.ndarray): x coordinate of input data points.
y	(numpy.ndarray): y coordinate of input data points.

Returns

r_sq (float): Coefficient of determination.

intercept (float): Intercept of the regression line.

slope (float): Slope of the regression line.

p_value (float): Two-sided p-value for a hypothesis test whose null hypothesis is that the slope is zero, using Wald Test with t-distribution of the test statistic.

std_err (float): Standard error of the estimated gradient.

6.3.1.16 normal_distr()

```
np.ndarray anko.stats_util.normal_distr (
            np.ndarray x,
            float a,
            float x0,
            float sigma )
```

Calculate normal distribution of input array x.

Parameters

x	(numpy.ndarray): Input values.
a	(float): Overall normalization constant.
x0	(float): Mean.
sigma	(float): Standard deviation.

Returns

out (numpy.ndarray): Output array.

6.3.1.17 right_half_normal_distr()

```
np.ndarray anko.stats_util.right_half_normal_distr (
            np.ndarray x,
            float a,
            float x0,
            float sigma )
```

Calculate right-side half normal distribution of input array x.

Parameters

x	(numpy.ndarray): Input values.
a	(float): Overall normalization constant.
x0	(float): Mean.
sigma	(float): Standard deviation.

Returns

out (numpy.ndarray): Output array.

6.3.1.18 smoothness()

```
def anko.stats_util.smoothness (
            np.ndarray x,
            bool normalize = False )
```

Parameters

x	(numpy.↩ ndarray):
normalize	(bool):

Returns

sm (numpy.ndarray):

6.3.1.19 three_stair_erf()

```
np.ndarray anko.stats_util.three_stair_erf (
            np.ndarray x,
            float c0,
            float c1,
            float c2,
            float x1,
            float x2 )
```

Parameters

x	(numpy.ndarray): Input values.
c0	(float):
c1	(float):
c2	(float):
x1	(float):
x2	(float):

Returns

out (numpy.ndarray): Output array.

6.3.1.20 z_normalization()

```
np.ndarray anko.stats_util.z_normalization (
            np.ndarray x )
```

Perform z-score normalizaion on input array x.

Parameters

x	(numpy.ndarray): Input values.

Returns

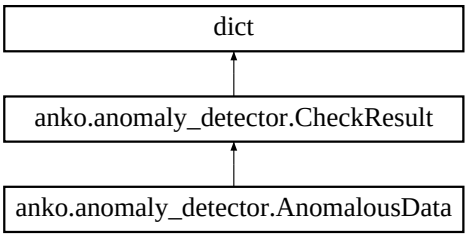
normalized_x (numpy.ndarray):

Chapter 7

Class Documentation

7.1 anko.anomaly_detector.AnomalousData Class Reference

Inheritance diagram for anko.anomaly_detector.AnomalousData:



Public Member Functions

- def __init__ (self)

7.1.1 Constructor & Destructor Documentation

7.1.1.1 __init__()

```
def anko.anomaly_detector.AnomalousData.__init__ (
              self )
```

The documentation for this class was generated from the following file:

- anko/anomaly_detector.py

7.2 anko.anomaly_detector.AnomalyDetector Class Reference

Public Member Functions

- def __init__ (self, t, series)
- object check (self)

Public Attributes

- apply_policies

  Policies for AnomalyDetector to follow with.
- t
- series
- check_failed
- thres_params

  Threshold values for selecting anomalous data.
- error_code
- models

  Models that can be considered by AnomalyDetector.

7.2.1 Detailed Description

Parameters

t	(array_↩ like):
series	(array_↩ like):

7.2.2 Constructor & Destructor Documentation

7.2.2.1 __init__()

```
def anko.anomaly_detector.AnomalyDetector.__init__ (
            self,
            t,
            series )
```

7.2.3 Member Function Documentation

7.2.3.1 check()

```
object anko.anomaly_detector.AnomalyDetector.check (
              self )
```

Returns

statsdata (dict):

7.2.4 Member Data Documentation

7.2.4.1 apply_policies

```
anko.anomaly_detector.AnomalyDetector.apply_policies
```

Policies for AnomalyDetector to follow with.

Parameters

scaleless_t	(bool)↩ :
boxcox	(bool)↩ :
z_normalization	(bool)↩ :
info_criterion	(str):
min_sample_size	(int):

7.2.4.2 check_failed

```
anko.anomaly_detector.AnomalyDetector.check_failed
```

7.2.4.3 error_code

```
anko.anomaly_detector.AnomalyDetector.error_code
```

7.2.4.4 models

```
anko.anomaly_detector.AnomalyDetector.models
```

Models that can be considered by AnomalyDetector.

Parameters

gaussian	(bool)↩ :
half_gaussian	(bool)↩ :
linear_regression	(bool)↩ :
step_func	(bool)↩ :
exp_decay	(bool)↩ :

7.2.4.5 series

`anko.anomaly_detector.AnomalyDetector.series`

7.2.4.6 t

`anko.anomaly_detector.AnomalyDetector.t`

7.2.4.7 thres_params

`anko.anomaly_detector.AnomalyDetector.thres_params`

Threshold values for selecting anomalous data.

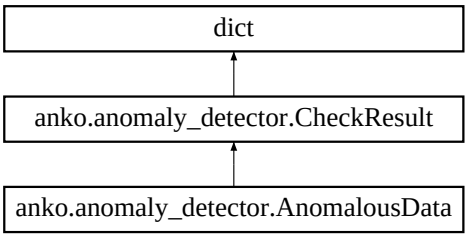
Parameters

The documentation for this class was generated from the following file:

- anko/anomaly_detector.py

7.3 anko.anomaly_detector.CheckResult Class Reference

Inheritance diagram for anko.anomaly_detector.CheckResult:



Public Member Functions

- def __getattr__ (self, name)
- def __repr__ (self)
- def __dir__ (self)

7.3.1 Member Function Documentation

7.3.1.1 __dir__()

```
def anko.anomaly_detector.CheckResult.__dir__ (
              self )
```

7.3.1.2 __getattr__()

```
def anko.anomaly_detector.CheckResult.__getattr__ (
              self,
              name )
```

7.3.1.3 __repr__()

```
def anko.anomaly_detector.CheckResult.__repr__ (
              self )
```

The documentation for this class was generated from the following file:

- anko/anomaly_detector.py

Chapter 8

File Documentation

8.1 anko/__init__.py File Reference

Namespaces

- anko

8.2 anko/anomaly_detector.py File Reference

Classes

- class anko.anomaly_detector.AnomalyDetector
- class anko.anomaly_detector.CheckResult
- class anko.anomaly_detector.AnomalousData

Namespaces

- anko.anomaly_detector

8.3 anko/stats_util.py File Reference

Namespaces

- anko.stats_util

Functions

- def anko.stats_util.get_histogram (np.ndarray x, bool sort_histo=False)

  Return the corresponding histogram of the data x.
- np.ndarray anko.stats_util.normal_distr (np.ndarray x, float a, float x0, float sigma)

  Calculate normal distribution of input array x.
- def anko.stats_util.gaussian_fit (np.ndarray x, bool sort_histo=False, str half=None, int maxfev=2000, bounds=[0, 1e+6])

  Fitting the Gaussian (normal) distribution for input data x.
- np.ndarray anko.stats_util.left_half_normal_distr (np.ndarray x, float a, float x0, float sigma)

  Calculate left-side half normal distribution of input array x.
- np.ndarray anko.stats_util.right_half_normal_distr (np.ndarray x, float a, float x0, float sigma)

  Calculate right-side half normal distribution of input array x.
- def anko.stats_util.flat_histogram (np.ndarray x)

  Manually assign parameters of Gaussian distrinution if the given histogram is too flat.
- def anko.stats_util.linear_regression (np.ndarray x, np.ndarray y)

  Fitting linear ansatz for input data (x, y).
- bool anko.stats_util.data_is_linear (np.ndarray x, np.ndarray y, float std_err_th=1e-2)

  Check whether the data (x, y) is linear under the given tolerance.
- np.ndarray anko.stats_util.general_erf (np.ndarray x, float a, float b, float x0)

  Calculate the generalize error function of input array x.
- np.ndarray anko.stats_util.three_stair_erf (np.ndarray x, float c0, float c1, float c2, float x1, float x2)
- def anko.stats_util.general_erf_fit (np.ndarray x, np.ndarray y, bool three_stair=False, int maxfev=2000, bounds=[0, 1e+6])

  Fitting generalize error function for input data (x, y).
- np.ndarray anko.stats_util.exp_decay (np.ndarray x, float a, float alpha)

  Calculate the exponential function of input array x.
- def anko.stats_util.exp_decay_fit (np.ndarray x, np.ndarray y, str mode='log-linregress', int maxfev=2000, bounds=[-1e-6, 1e+6])
- def anko.stats_util.smoothness (np.ndarray x, bool normalize=False)
- def anko.stats_util.discontinuous_idx (np.ndarray x, int std_width=1)
- bool anko.stats_util.is_oscillating (np.ndarray x, float osci_freq_th=0.3)

  Determine whether the input array x is oscillating over its mean with frequency larger than osci_freq_th.
- np.ndarray anko.stats_util.fitting_residual (np.ndarray x, np.ndarray y, func, args, float mask_min=None, bool absolute_value=True, bool standardized=False)

  Compute the fitting residual.
- float anko.stats_util.AIC_score (np.ndarray y, np.ndarray y_predict, int p)

  Compute Akaike information criterion for model selection.
- float anko.stats_util.BIC_score (np.ndarray y, np.ndarray y_predict, int p)

  Compute Bayesian information criterion for model selection.
- np.ndarray anko.stats_util.z_normalization (np.ndarray x)

  Perform z-score normalizaion on input array x.

8.4 README.md File Reference