Least Squares Anomaly Detection

Release 1.4

Jan 06, 2020

Contents:

LSANOMALY

least squares anomaly detection

lsanomaly is a flexible, fast, probabilistic method for calculating outlier scores on test data, given training examples of inliers. Out of the box it works well with *scikit-learn* packages. See the features section for why you might chose this model over other options.

CHAPTER 1

Features

- Compatible with scikit-learn package modules
- Probabilistic outlier detection model
- Robust classifier when given multiple inlier classes
- Easy to install and get started

CHAPTER 2

Installation

The best way to install lsanomaly is to:

```
pip install lsanomaly
```

An alternative is to download the source code and

```
python setup.py install
```

Tests can be run from *setup* if *pytest* is installed:

```
python setup.py test
```

CHAPTER 3

Usage

For those familiar with scikit-learn the interface will be familiar, in fact *lsanomaly* was built to be compatible with *sklearn* modules where applicable. Here is basic usage of *lsanomaly* to get started quick as possible.

Configuring the Model

LSAD provides reasonable default parameters when given an empty init or it can be passed values for *rho* and *sigma*. The value rho controls sensitivity to outliers and sigma determines the 'smoothness' of the boundary. These values can be tuned to improve your results using lsanomaly.

```
from lsanomaly import LSAnomaly

# At train time lsanomaly calculates parameters rho and sigma
lsanomaly = LSAnomaly()
# or
lsanomaly = LSAnomaly(sigma=3, rho=0.1, seed=42)
```

Training the Model

After the model is configured the training data can be fit.

```
import numpy as np
lsanomaly = LSAnomaly(sigma=3, rho=0.1, seed=42)
lsanomaly.fit(np.array([[1],[2],[3],[1],[2],[3]]))
```

Making Predictions

Now that the data is fit, we will probably want to try and predict on some data not in the training set.

```
>>> lsanomaly.predict(np.array([[0]]))
[0.0]
>>> lsanomaly.predict_proba(np.array([[0]]))
array([[ 0.7231233,  0.2768767]])
```

CHAPTER 4

Documentation

Full documentation can be built using Sphinx.

CHAPTER 5

Examples

See *notebooks/* for sample applications.

CHAPTER 6

Reference

J.A. Quinn, M. Sugiyama. A least-squares approach to anomaly detection in static and sequential data. Pattern Recognition Letters 40:36-40, 2014.

[pdf]

CHAPTER 7

Least Squares Anomaly Detection API

7.1 Indices and tables

- genindex
- modindex
- search

7.2 Least Squares Anomaly Detection

class lsanomaly._lsanomaly.**LSAnomaly**(*n_kernels_max=500*, *kernel_pos=None*, *sigma=None*, *rho=None*, *gamma=None*, *seed=None*)

decision_function(*X*)

Generate an inlier score for each test data example.

Args X (numpy.ndarray): Test data, of dimension N times d (rows are examples, columns are data dimensions)

Returns: numpy.ndarray: A vector of length N, where each element contains an inlier score in the range 0-1 (outliers have values close to zero, inliers have values close to one).

fit(*X*, *y=None*, *k=5*)

Fit the inlier model given training data. This function attempts to choose reasonable defaults for parameters sigma and rho if none are specified, which could then be adjusted to improve performance.

Args: X (numpy.ndarray): Examples of inlier data, of dimension N times d (rows are examples, columns are data dimensions)

y (numpy.ndarray): If the inliers have multiple classes, then *y* contains the class assignments as a vector of length N. If this is specified then the model will attempt to assign test data to one of the inlier classes or to the outlier class.

k (int): Number of nearest neighbors to use in the KNN kernel length scale heuristic.

Returns: self

get_params (*deep=True*)
Not implemented.

Args: deep (bool):

Returns:

predict (*X*)
Assign classes to test data.

Args: X (numpy.ndarray): Test data, of dimension N times d (rows are examples, columns are data dimensions)

Returns: numpy.ndarray:

A vector of length N containing assigned classes. If no inlier classes were specified during training, then 0 denotes an inlier and 1 denotes an outlier. If multiple inlier classes were specified, then each element of y_predicted is either one of those inlier classes, or an outlier class (denoted by the maximum inlier class ID plus 1).

predict_proba (*X*)
Calculate posterior probabilities of each inlier class and the outlier class for test data.

Args X (numpy.ndarray): Test data, of dimension N times d (rows are examples, columns are data dimensions)

Returns numpy.ndarray: An array of dimension N times n_inlier_classes+1, containing the probabilities of each row of X being one of the inlier classes, or the outlier class (last column).

predict_sequence (*X*, *A*, *pi*, *inference='smoothing'*)
Calculate class probabilities for a sequence of data.

Args X (numpy.ndarray): Test data, of dimension N times d (rows are time frames, columns are data dimensions)

A (numpy.ndarray):: Class transition matrix, where A[i,j] contains p(y_t=jly_{t-1}=i)

pi (numpy.ndarray): vector of initial class probabilities

inference (str) : 'smoothing' or 'filtering'.

Returns numpy.ndarray: An array of dimension N times n_inlier_classes+1, containing the probabilities of each row of X being one of the inlier classes, or the outlier class (last column).

score (*X*, *y*)
Calculate accuracy score, needed because of bug in metrics.accuracy_score when comparing list with numpy array.

set_params (***params*)
Not implemented.

Args: **params (dict):

7.3 Kernel Length Scale Approximation

lsanomaly.lengthscale_approx.**median_kneighbour_distance** (*X*, *k=5*, *seed=None*)
Calculate the median distance between a set of random data points and their kth nearest neighbours. This is a heuristic for setting the kernel length scale.

Args: X (numpy.ndarray): Data points k (int): Number of neighbors to use seed (int): random number seed

Returns: float: Kernel length scale estimate

Raises: ValueError: If the number of requested neighbors *k* is less than the number of observations in *X*.

lsanomaly.lengthscale_approx.**pair_distance_centile**(*X*, *centile*, *max_pairs=5000*, *seed=None*)

Calculate centiles of distances between random pairs in a data-set. This an alternative to the median kNN distance for setting the kernel length scale.

Args: X (numpy.ndarray): Data observations centile (int): distance centile max_pairs (int): maximum number of pairs to consider seed (int): random number seed

Returns: float: length scale estimate

CHAPTER 8

Evaluating LSAnomaly

Scripts for evaluating *lsanomaly* against other methods is provided in J Quinn's software https://cit.mak.ac.ug/staff/jquinn/software/lsanomaly.html. Owing to changes in APIs and availability of some test data, that code has been refactored and expanded.

There are three commandline applications that will be download the test data, perform a 5-fold cross-validation and, produce a LaTeX document summarizing the results. Each of the three applications has a *main* method that can be used for further automation.

The following datasets are configured for download (see *evaluate/eval_params.yml*) from https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/.

- australian
- breast-cancer
- cod-rna
- colon-cancer.bz2
- diabetes
- dna.scale
- glass.scale
- heart
- ionosphere_scale
- letter.scale
- leu.bz2
- mnist.bz2
- mushrooms
- pendigits
- satimage.scale

- sonar_scale

Those with an extension of *.bz2* will be inflated. The compressed version is retained.

8.1 Downloading Test Data

download.py

A commandline utility to retrieve test data from https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/ for use in evaluating LSAnamoly.

usage: download.py [-h] –params YML_PARAMS –data-dir DATA_DIR [–sc-url SC_URL] [–mc-url MC_URL]

Retrieve datasets for LsAnomaly evaluation. By default, data is retrieved from https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/

Arguments

-h, --help show this help message and exit

--params YML_PARAMS, -p YML_PARAMS YAML file with evaluation parameters

--data-dir DATA_DIR, -d DATA_DIR directory to store retrieved data sets

--sc-url SC_URL optional: single class test data URL; default: https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary/

--mc-url MC_URL optional: Multi-class test data URL; default: https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multiclass/

`lsanomaly.evaluate.download.`**`main`**(*param_file*, *sc_url*, *mc_url*, *data_fp*)

The main show. Tries to retrieve and store all the configured data-sets.

Args: param_file (str): *.yml* File containing the evaluation parameters

sc_url (str): single class data set URL

mc_url (str): multiclass data set URL

data_fp (str): Directory where the datasets will be written

Raises: ValueError: If *data_fp* is not a valid directory.

`lsanomaly.evaluate.download.`**`unzip_write`**(*file_path*)

Reads and inflates a .bz2 file and writes it back. The compressed file is retrained. Used internally.

Args: file_path (str): file to inflate

Raises: FileNotFoundError

`lsanomaly.evaluate.download.`**`write_contents`**(*file_path*, *get_request*)

Writes the contents of the get request to the specified file path.

Args: file_path (str): file path

get_request (requests.Response): response object

Raises: IOError

`lsanomaly.evaluate.download.`**`get_request`**(*dataset*, *file_path*, *sc_url*, *mc_url*)

Retrieve *dataset* trying first at *sc_url* and failing that, at *mc_url*. If a data set cannot be retrieved, it is skipped. The contents to *file_path* with the data set name as the file name.

Args: dataset (str): Dataset name as referenced in https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/

file_path (str): Directory where *dataset* will be written.

sc_url (str): single class data set URL

mc_url (str): multiclass data set URL

8.2 Evaluating the Test Data

run_eval.py

Least squares anomaly evaluation on static data. After running experiments, use *generate_latex.py* to create a table of results.

This is a refactored and updated version of the script in *evaluate_lsanomaly.zip* (see https://cit.mak.ac.ug/staff/jquinn/software/lsanomaly.html).

usage: run_eval.py [-h] –data-dir DATA_DIR –output-json JSON_FILE

Perform evaluation of LSAnomaly on downloaded data-sets. 5-fold cross validation is performed.

Arguments

-h, --help show this help message and exit

--data-dir DATA_DIR, -d DATA_DIR directory of stored data-sets in *libsvm* format

--params YML_PARAMS, -p YML_PARAMS YAML file with evaluation parameters

--output-json JSON_FILE, -o JSON_FILE path and file name of the results

`lsanomaly.evaluate.run_eval.`**`evaluate`**(*X_train*, *y_train*, *X_test*, *y_test*, *outlier_class*, *method_name*, *current_method_aucs*, *sigma*, *rho=0.1*, *nu=0.5*)

Evaluation for a method and data set. Calculates the AUC for a single evaluation fold.

Args: X_train (numpy.ndarray): independent training variables

y_train (numpy.ndarray): training labels

X_test (numpy.ndarray): independent test variables

y_test (numpy.ndarray): test labels

outlier_class (int): index of the outlier class

method_name (str): method being run

current_method_aucs (list): input to the *results* dictionary

sigma (float): kernel lengthscale for LSAD and OCSVM

rho (float): smoothness parameter for LSAD

nu (float): OCSVM parameter - see *scikit-learn* documentation

Raises: ValueError: if a *NaN* is encountered in the AUC calculation.

`lsanomaly.evaluate.run_eval.`**`gen_data`**(*data_sets*)

Generator to deliver independent, dependent variables and the name of the data set.

Args: data_sets (list): data sets read from the data directory

Returns: numpy.ndarray, numpy.ndarray, str: *X*, *y*, *name*

lsanomaly.evaluate.run_eval.**gen_dataset**(*data_dir*)
Generator for the test data file paths. All test files must be capable of being loaded by *load_svmlight_file()*. Files with extensions *.bz2*, *.csv* are ignored. Any file beginning with . is also ignored.

This walks the directory tree starting at *data_dir*, therefore all subdirectories will be read.

Args: data_dir (str): Fully qualified path to the data directory

Returns: list: *svmlight* formatted data sets in *data_dir*

lsanomaly.evaluate.run_eval.**main**(*data_dir*, *json_out*, *param_file*, *n_splits=5*, *rho=0.1*, *nu=0.5*)
The main show. Loop through all the data-sets and methods running a 5-fold stratified cross validation. The results are saved to the specified *json_out* file for further processing.

Args: data_dir (str): directory holding the downloaded data sets

json_out (str): path and filename to store the evaluation results

param_file (str): YAML file with evaluation parameters

n_splits (int): number of folds in the cross-validation.

8.3 Create a Results Table in LaTeX

generate_latex.py

A commandline application to create a latex table summarising the results of LSAnomaly static data experiments. This is a refactored version of the script in *evaluate_lsanomaly.zip* (see https://cit.mak.ac.ug/staff/jquinn/software/lsanomaly.html).

usage

generate_latex.py [-h] –input-json JSON_FILE –latex-output LATEX_FILE

Create a LaTeX document with a table of results

Arguments

-h, --help show this help message and exit

--input-json JSON_FILE, -i JSON_FILE path and file name of the results

--latex-output LATEX_FILE, -o LATEX_FILE path and file name of the LaTeX file

lsanomaly.evaluate.generate_latex.**results_table**(*json_results*)
Build the LaTeX table.

Args: json_results (dict): results from *run_eval*

Returns: str: LaTeX table

lsanomaly.evaluate.generate_latex.**main**(*input_json*, *output_latex*)
Read the JSON results file; generate the table in LaTeX; wrap the results table in a simple LaTeX document and write it to *output_latex*

Args: input_json (str): file of the JSON-serialized results

output_latex (str): file where the LaTeX document will be written

CHAPTER 9

License

The MIT License (MIT)

Copyright (c) 2016 John Quinn

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Python Module Index

l

Index