Package: oddstream (via r-universe)

November 21, 2024

Type Package

Title Outlier Detection in Data Streams

Version 0.5.1

Depends R (>= 3.4.0)

Maintainer Priyanga Dilini Talagala <pritalagala@gmail.com>

Description We proposes a framework that provides real time support for early detection of anomalous series within a large collection of streaming time series data. By definition, anomalies are rare in comparison to a system's typical behaviour. We define an anomaly as an observation that is very unlikely given the forecast distribution. The algorithm first forecasts a boundary for the system's typical behaviour using a representative sample of the typical behaviour of the system. An approach based on extreme value theory is used for this boundary prediction process. Then a sliding window is used to test for anomalous series within the newly arrived collection of series. Feature based representation of time series is used as the input to the model. To cope with concept drift, the forecast boundary for the system's typical behaviour is updated periodically. More details regarding the algorithm can be found in Talagala, P. D., Hyndman, R. J., Smith-Miles, K., et al. (2019) <doi:10.1080/10618600.2019.1617160>.

BugReports https://github.com/pridiltal/oddstream/issues

License GPL-3

LazyData true

RoxygenNote 7.1.0

Imports pcaPP, stats, ggplot2, ks, MASS, RcppRoll, mgcv, moments , RColorBrewer, mvtsplot, tibble, reshape, dplyr, graphics, tidyr, kernlab, magrittr

Encoding UTF-8

Suggests testthat, tidyverse

Config/pak/sysreqs libicu-dev

Repository https://robjhyndman.r-universe.dev

RemoteUrl https://github.com/pridiltal/oddstream

RemoteRef HEAD

RemoteSha acb95325e27b475fb6c91151cf6f3d18ca622563

Contents

`anomalous_stream` *Multivariate timeseries dataset with an anomalous event.*

Description

A mutivariate time series dataset with some anomalous series. These time series are with noisy signals.

Usage

```
anomalous_stream
```

Format

A data frame with 640 series each with 1459 time points.

`extract_tsfeatures` *Extract features from a collection of time series*

Description

This function extract time series features from a collection of time series. This is a modification oftsmeasures function of anomalous package package .

Usage

```
extract_tsfeatures(
  y,
  normalise = TRUE,
  width = ifelse(frequency(y) > 1, frequency(y), 10),
  window = width
)
```

Arguments

`y`	A multivariate time serie
`normalise`	If TRUE, each time series is scaled to be normally distributed with mean 0 and sd 1
`width`	A window size for variance change, level shift and lumpiness
`window`	A window size for KLscore

Value

An object of class features with the following components:

`mean`	Mean
`variance`	Variance
`lumpiness`	Variance of annual variances of remainder
`lshift`	Level shift using rolling window
`vchange`	Variance change
`linearity`	Strength of linearity
`curvature`	Strength of curvature
`spikiness`	Strength of spikiness
`season`	Strength of seasonality
`peak`	Strength of peaks
`trough`	Strength of trough
`BurstinessFF`	Burstiness of time series using Fano Factor
`minimum`	Minimum value
`maximum`	Maximum value
`rmeaniqmean`	Ratio between interquartile mean and the arithmetic mean
`moment3`	Third moment
`highlowmu`	Ratio between the means of data that is below and upper the global mean

References

Hyndman, R. J., Wang, E., & Laptev, N. (2015). Large-scale unusual time series detection. In 2015 IEEE International Conference on Data Mining Workshop (ICDMW), (pp. 1616-1619). IEEE.

Fulcher, B. D. (2012). Highly comparative time-series analysis. PhD thesis, University of Oxford.

See Also

find_odd_streams, get_pc_space, set_outlier_threshold, gg_featurespace

Examples

```
mvtsplot::mvtsplot(anomalous_stream, levels=8, gcol=2, norm="global")
features <- extract_tsfeatures(anomalous_stream[500:550, ])
plot.ts(features[, 1:10])
```

---

find_odd_streams *Detect outlying series within a collection of sreaming time series*

Description

This function detect outlying series within a collection of streaming time series. A sliding window is used to handle straming data. In the precence of concept drift, the forecast boundary for the system's typical behaviour can be updated periodically.

Usage

```
find_odd_streams(
  train_data,
  test_stream,
  update_threshold = TRUE,
  window_length = nrow(train_data),
  window_skip = window_length,
  concept_drift = FALSE,
  trials = 500,
  p_rate = 0.001,
  cd_alpha = 0.05
)
```

Arguments

train_data	A multivariate time series data set that represents the typical behaviour of the system.
test_stream	A multivariate streaming time series data set to be tested for outliers
update_threshold	If TRUE, the threshold value to determine outlying series is updated. The default value is set to TRUE
window_length	Sliding window size (Ideally this window length should be equal to the length of the training multivariate time series data set that is used to define the outlying threshold)

window_skip	The number of steps the window should slide forward. The default is set to window_length
concept_drift	If TRUE, The outlying threshold will be updated after each window. The default is set to FALSE
trials	Input for set_outlier_threshold function. Default value is set to 500.
p_rate	False positive rate. Default value is set to 0.001.
cd_alpha	Singnificance level for the test of non-stationarity.

Value

a list with components

out_marix	The indices of the outlying series in each window
p_value	p-value for the two sample comparison test for concept drift detection
anom_threshold	anomalous threshold

For each window a plot is also produced on the current graphic device

References

Clifton, D. A., Hugueny, S., & Tarassenko, L. (2011). Novelty detection with multivariate extreme value statistics. Journal of signal processing systems, 65 (3),371-389.

Duong, T., Goud, B. & Schauer, K. (2012) Closed-form density-based framework for automatic detection of cellular morphology changes. PNAS, 109, 8382-8387.

Talagala, P., Hyndman, R., Smith-Miles, K., Kandanaarachchi, S., & Munoz, M. (2018). Anomaly detection in streaming nonstationary temporal data (No. 4/18). Monash University, Department of Econometrics and Business Statistics.

See Also

extract_tsfeatures, get_pc_space, set_outlier_threshold, gg_featurespace

Examples

```
#Generate training dataset
set.seed(890)
nobs = 250
nts = 100
train_data <- ts(apply(matrix(ncol = nts, nrow = nobs), 2, function(nobs){10 + rnorm(nobs, 0, 3)}))
# Generate test stream with some outliying series
nobs = 15000
test_stream <- ts(apply(matrix(ncol = nts, nrow = nobs), 2, function(nobs){10 + rnorm(nobs, 0, 3)}))
test_stream[360:1060, 20:25] = test_stream[360:1060, 20:25] * 1.75
test_stream[2550:3550, 20:25] =  test_stream[2550:3550, 20:25] * 2
find_odd_streams(train_data, test_stream , trials = 100)


# Considers the first window  of the data set as the training set and the remaining as
# the test stream
```

```
train_1data <- anomalous_stream[1:100,]
test_stream <-anomalous_stream[101:1456,]
find_odd_streams(train_data, test_stream , trials = 100)
```

---

get_pc_space *Define a feature space using the PCA components of the feature matrix*

Description

Define a two dimensional feature space using the first two principal components generated from the fetures matrix returned by `extract_tsfeatures`

Usage

```
get_pc_space(features, robust = TRUE, kpc = 2)
```

Arguments

`features`	Feature matrix returned by `extract_tsfeatures`
`robust`	If TRUE, a robust PCA will be used on the feature matrix.
`kpc`	Desired number of components to return.

Value

It returns a list with class 'pcattributes' containing the following components:

`pcnorm`	The scores of the firt kpc pricipal components
`center, scale`	The centering and scaling used
`rotation`	the matrix of variable loadings (i.e., a matrix whose columns contain the eigenvectors). The function `princomp` returns this in the element loadings.

See Also

`PCAproj`, `prcomp`, `find_odd_streams`, `extract_tsfeatures`, `set_outlier_threshold`, `gg_featurespace`

Examples

```
features <- extract_tsfeatures(anomalous_stream[1:100, 1:100])
pc <- get_pc_space(features)
```

gg_featurespace — *Produces a ggplot object of two dimensional feature space.*

Description

Create a ggplot object of two dimensional feature space using the first two pricipal component returned by `get_pc_space`.

Usage

```
gg_featurespace(object, ...)
```

Arguments

`object`	Object of class "`pcoddstream`".
`...`	Other plotting parameters to affect the plot.

Value

A ggplot object of two dimensional feature space.

See Also

`find_odd_streams`, `extract_tsfeatures`, `get_pc_space`, `set_outlier_threshold`

Examples

```
features <- extract_tsfeatures(anomalous_stream[1:100, 1:100])
pc <- get_pc_space(features)
p <- gg_featurespace(pc)
p + ggplot2::geom_density_2d()
```

oddstream — *oddstream: A package for Outlier Detection in Data Streams*

Description

Rapid advances in hardware technology have enabled a wide range of physical objects, living beings and environments to be monitored using sensors attached to them. Over time these sensors generate streams of time series data. Finding anomalous events in streaming time series data has become an interesting research topic due to its wide range of possible applications such as: intrusion detection, water contamination monitoring, machine health monitoring, etc. This package proposes a framework that provides real time support for early detection of anomalous series within a large collection of streaming time series data. By definition, anomalies are rare in comparison to a system's typical behaviour. We define an anomaly as an observation that is very unlikely given the forecast distribution. The proposed framework first forecasts a boundary for the system's typical behaviour using a representative sample of the typical behaviour of the system. An approach based on extreme value theory is used for this boundary prediction process. Then a sliding window is used to test for anomalous series within the newly arrived collection of series. Feature based representation of time series is used as the input to the model. To cope with concept drift, the forecast boundary for the system's typical behaviour is updated periodically. More details regarding the algorithm can be found in Talagala, P. D., Hyndman, R. J., Smith-Miles, K., et al. (2019) DOI:10.1080/10618600.2019.1617160.

Note

The name `oddstream` comes from `Outlier Detection in Data STREAMs`

References

Clifton, D. A., Hugueny, S., & Tarassenko, L. (2011). Novelty detection with multivariate extreme value statistics. Journal of signal processing systems, 65 (3),371-389.

Talagala, P. D., Hyndman, R. J., Smith-Miles, K., et al. (2019). Anomaly detection in streaming nonstationary temporal data. Journal of Computational and Graphical Statistics, 1-28. DOI:10.1080/10618600.2019.1617160

See Also

The core functions in this package: `find_odd_streams`, `extract_tsfeatures`, `get_pc_space`, `set_outlier_threshold`, `gg_featurespace`

`set_outlier_threshold` *Set a threshold for outlier detection*

Description

This function forecasts a boundary for the typical behaviour using a representative sample of the typical behaviour of a given system. An approach based on extreme value theory is used for this boundary prediction process.

Usage

```
set_outlier_threshold(pc_pcnorm, p_rate = 0.001, trials = 500)
```

Arguments

`pc_pcnorm`	The scores of the first two pricipal components returned by `get_pc_space`
`p_rate`	False positive rate. Default value is set to 0.001
`trials`	Number of trials to generate the extreme value distirbution. Default value is set to 500.

Value

Returns a threshold to determine outlying series in the next window consists with a collection of time series.

References

Clifton, D. A., Hugueny, S., & Tarassenko, L. (2011). Novelty detection with multivariate extreme value statistics. Journal of signal processing systems, 65 (3),371-389.

Talagala, P., Hyndman, R., Smith-Miles, K., Kandanaarachchi, S., & Munoz, M. (2018). Anomaly detection in streaming nonstationary temporal data (No. 4/18). Monash University, Department of Econometrics and Business Statistics.

See Also

find_odd_streams, extract_tsfeatures, get_pc_space, gg_featurespace

Examples

```
# Generate training dataset
set.seed(123)
nobs <- 500
nts <- 50
train_data <- ts(apply(matrix(ncol = nts, nrow = nobs), 2, function(nobs){10 + rnorm(nobs, 0, 3)}))
features <- extract_tsfeatures(train_data)
pc <- get_pc_space(features)
threshold <- set_outlier_threshold(pc$pcnorm)
threshold$threshold_fnx
```

Index