Package 'mvMonitoring'

January 31, 2017

Type Package

Title Multi-State Adaptive Dynamic Principle Components Analysis for Multivariate Process Monitoring

Version 0.1.0

Author Gabriel Odom, Ben Barnard, and Karen Kazor

Maintainer Gabriel Odom <`gabriel_odom@baylor.edu`>

Description Use multi-state splitting to apply Adaptive-Dynamic PCA to data generated from a continuous-time multivariate industrial or natural process. Employ PCA-based dimension reduction to extract linear combinations of relevant features, reducing computational burdens.

License What license is it under?

Depends BMS, lazyeval, zoo, xts, utils

Encoding UTF-8

LazyData true

RoxygenNote 5.0.1

Suggests knitr,
rmarkdown

VignetteBuilder knitr

R topics documented:

`faultDetect` *Process Fault Detection*

Description

Detect if a single multivariate observation is beyond normal parameters.

Usage

```
faultDetect(threshold_object, observation, ...)
```

Arguments

`threshold_object`	An object of classes "threshold" and "pca" returned by the internal threshold() function.
`observation`	A single row of an xts data matrix to compare against the thresholds
`...`	Lazy dots for additional internal arguments

Details

This function takes in a threshold object returned by the threshold() function and a single observation as a matrix or xts row. The function then returns a row vector of the SPE test statistics, a logical indicator marking if this statistic is beyond the threshold, the Hotelling's T2 statistic, and an indicator if this statistic is beyond the threshold. Observations with monitoring statistics beyond the calculated threshold are marked with a 1, while within-parameter observations are marked with a 0. These threshold values are passed from the threshold() function through this function via a returned threshold object. This object will be used in higher function calls.

This internal function is called by faultFilter().

Value

A named 1 * 4 matrix of the SPE statistic value ("SPE"), SPE fault indicator ("SPE_Flag"), T2 statistic value ("T2"), and T2 fault indicator for the single row observation passed to this function ("T2_Flag").

`faultFilter` *Process Fault Filtering*

Description

Flag and filter out observations beyond normal parameters, then return the observations within normal operating conditions.

Usage

```
faultFilter(trainData, testData, updateFreq, faultsToTriggerAlarm, ...)
```

Arguments

`trainData`	An xts data matrix of initial training observations
`testData`	The data not included in the training data set
`updateFreq`	How many observations from the test data matrix will be returned to update the training data matrix and move it forward?
`faultsToTriggerAlarm`	Specifies how many sequential faults will cause an alarm to trigger. Defaults to 3.
`...`	Lazy dots for additional internal arguments

Details

This function is essentially a wrapper function to call and organize the output from these other internal functions: faultDetect(), threshold(), and pca(). It is applied over a rolling window, with observation width equal to updateFreq, of the larger full data matrix via the processMonitor() function, wherein the testing and training data sets move forward in time across the entire data matrix.

This internal function is called by processMonitor().

Value

A list of class "fault_ls" with the following: faultObj = an xts flagging matrix with the same number of rows as "testData". This flag matrix has the following five columns: the SPE test statistic for each observation in "testData", an SPE indicator recording 0 if the test statistic is less than or equal to the critical value passed through from the threshold object, a T2 test statistic, a similar T2 indicator, and a final column indicating if there have been three flags in a row for either the SPE or T2 monitoring statistics or both. nonAlarmedTestObs = an xts matrix of all the rows of the training data which were not alarmed. trainSpecs = the threshold object returned by the internal threshold() function. See this function’s help file for more details.

`mspMonitor` *Real-Time Process Monitoring Function*

Description

Monitor and flag (if necessary) incoming multivariate process observations.

Usage

```
mspMonitor(observations, labelVector, trainingSummary, ...)
```

Arguments

`observations`	an n * p xts matrix. For real-time monitoring via a script within a batch file, n = 1, so this must be a matrix of a single row.
`labelVector`	an n * 1 integer vector of class memberships
`trainingSummary`	the TrainingSpecs list returned by the mspTrain() function. This list contains –for each class– the SPE and T2 thresholds, as well the projection matrix.
`...`	Lazy dots for additional internal arguments

Details

This function is designed to be ran at specific time intervals (every 10 seconds, 30 seconds, 1 minute, 5 minutes, 10 minutes, etc), from a batch file hosted script which calls this function and mspWarning(). This function takes in the specific observations to monitor and their class memberships (if any) and returns an xts matrix of these observations column concatenated with their monitoring statistic values, flag statuses, and an empty alarm column. Users should then append these rows onto a previously existing daily observations matrix. The mspWarning() function will then take in the daily observation xts matrix with updated rows returned by this function and check the monitoring statistic flag indicators to see if an alarm status has been reached. For further details, see the mspWarning() function.

This function calls the faultDetect() function, and requires the training information returned by the mspTrain function. This function will return the xts matrix necessary for the mspWarning() function.

Value

An n * (p + 5) xts matrix, where the last five columns are the monitoring statistics and corresponding fault flags, and an empty alarm column

mspTrain *Multi-State Adaptive-Dynamic Process Training*

Description

This function performs Multi-State Adaptive-Dynamic PCA on a data set with time-stamped observations.

Usage

```
mspTrain(data, labelVector, trainObs, updateFreq = ceiling(0.5 * trainObs),
  Dynamic = TRUE, lagsIncluded = 1, faultsToTriggerAlarm = 3, ...)
```

Arguments

`data`	An xts data matrix
`labelVector`	Class label vector (as logical or finite numeric)
`trainObs`	The number of observations upon which to train the algorithm
`updateFreq`	The algorithm update frequency (defaulting to half as many observations as the training frequency)
`Dynamic`	Should the PCA algorithm include lagged variables? Defaults to TRUE
`lagsIncluded`	If Dynamic = TRUE, how many lags should be included? Defaults to 1.
`faultsToTriggerAlarm`	The number of sequential faults needed to trigger an alarm
`...`	Lazy dots for additional internal arguments

Details

This function is designed to identify and sort out sequences of observations which fall outside normal operating conditions. We assume that the process data are time-dependent in both seasonal and non-stationary effects (which necessitate the Adaptive and Dynamic components, respectively). We further assume that this data is drawn from a multivariate process under multiple mutually exclusive states, implying that the linear dimension reduction projection matrices may be different for each state. Therefore, in summary, this function lags the features to account for correlation between sequential observations, splits the data by classes, and re-estimates projection matrices on a rolling window to account for seasonality. Further, this function uses non-parametric density estimation to calculate the 1 - alpha quantiles of the SPE and Hotelling's T2 statistics from a set of training observations, then flags any observation in the testing data set with process monitoring statistics beyond these calculated critical values. Becuase of natural variablity inherent in all real data, we do not remove observations simply because they are have been flagged as outside normal operating conditions. This function records an alarm only for observations having three flags in a row, as set by the default argument value of "faultsToTriggerAlarm". These alarm-positive observations are then removed from the data set and held in a separate xts matrix for inspection.

This user-facing function calls the processMonitor() function, and returns the training arguments necessary to call the mspMonitor() and mspWarning() functions.

Value

A list of the following components: FaultChecks = an xts data matrix containing the SPE monitoring statistic and corresponding logical flagging indicator, the Hotelling's T2 monitoring statitisic and corresponding logical flagging indicator, and the Alarm indicator. Non_Alarmed_Obs = an xts data matrix of all the non-Alarmed observations. Alarms = and an xts data matrix of the features and specific alarms for Alarmed observations, where the alarm code is as follows: 0 = no alarm, 1 = Hotelling's T2 alarm, 2 = Squared Prediction Error alarm, and 3 = both alarms. TrainingSpecs = a list of k lists, one for each class, with each list containing the specific threshold object returned by the internal threshold() function for that class. See this function's help file for more details.

`mspWarning` *Process Alarms*

Description

Trigger an alarm, if necessary, for incoming multivariate process observations.

Usage

```
mspWarning(mspMonitor_object, faultsToTriggerAlarm = 3)
```

Arguments

`mspMonitor_object`	An xts matrix returned by the mspMonitor() function
`faultsToTriggerAlarm`	Specifies how many sequential faults will cause an alarm to trigger. Defaults to 3.

Details

This function and the mspMonitor() function are designed to be ran via a script within a batch. The file flow is as follows: at each time interval, run the mspMonitor() function on the daily observation matrix to add a flag status to the most recent incoming observation in the matrix, and return this new xts matrix. Then, pass this updated daily observation matrix to the mspWarning() function, which will check if the process has recorded three or more sequential monitoring statistic flags in a row. Of note, since these functions are expected to be repeatedly ran in real time, this function will only check for an alarm within the last row of the xts matrix. To check multiple rows for an alarm state, please use the mspTrain function, which was designed to check multiple past observations.

This function requires an xts matrix returned by the mspMonitor() function.

Value

An xts matrix of the same dimensions as mspMonitor_object, with a recorded negative or positive and type-specific alarm status

---

`pca` *PCA for Data Scatter Matrix*

Description

Calculate the principal components analysis for a data matrix, and also find the squared prediction error (SPE) and Hotelling's T2 test statistic values for each observation in this data matrix.

Usage

```
pca(data, var.amnt = 0.95, ...)
```

Arguments

`data`	A centred-and-scaled data matrix or xts matrix
`var.amnt`	How much energy should be preserved in the projection? Defaults to 0.95.
`...`	Lazy dots for additional internal arguments

Details

This function takes in a training data matrix, without the label column, and the energy preservation proportion, which defaults to 95 percent per Kazor et al (2016). This proportion is the sum of the q largest eigenvalues divided by the sum of all p eigenvalues, where q is the number of columns of the p * q projection matrix P. This function then returns the projection matrix P, a diagonal matrix of the reciprocal eigenvalues (LambdaInv), a vector of the SPE test statistic values corresponding to the rows of the data matrix, and a T2 test statistic vector similar to the SPE vector.

This internal function is called by faultFilter().

Value

A list of class "pca" with the following: projectionMatrix = the q eigenvectors corresponding to the q largest eigenvalues as a p * q projection matrix. LambdaInv = the diagonal matrix of inverse eigenvalues. SPE = the vector of SPE test statistic values for each of the n observations contained in the data matrix. T2 = the vector of Hotelling's T2 test statistic for each of the same n observations.

processMonitor *Adaptive Process Training*

Description

Apply Adaptive-Dynamic PCA to state-specific data matrices.

Usage

```
processMonitor(data, trainObs, updateFreq = ceiling(0.2 * trainObs),
  faultsToTriggerAlarm = 3, ...)
```

Arguments

`data`	An xts data matrix
`trainObs`	How many train observations will be used
`updateFreq`	How many non-flagged rows to collect before the function updates
`faultsToTriggerAlarm`	the number of sequential faults needed to trigger an alarm
`...`	Lazy dots for additional internal arguments

Details

This function is the class-specific implementation of the Adaptive- Dynamic PCA described in the details of the mspTrain function. See that function's help file for further details.

This internal function is called by mspTrain(). This function calls the faultFilter() function.

Value

A list of the following components: FaultChecks = a class specific xts data matrix containing the SPE monitoring statistic and corresponding logical flagging indicator, the Hotelling's T2 monitoring statitisic and corresponding logical flagging indicator, and the Alarm indicator. Non_Alarmed_Obs = a class specific xts data matrix of all the observations with alarm states equal to 0. Alarms = a class-specific xts data matrix of the features and specific alarms for Alarmed observations, where the alarm code is as follows: 0 = no alarm, 1 = Hotelling's T2 alarm, 2 = Squared Prediction Error alarm, and 3 = both alarms. trainSpecs = the threshold object returned by the internal threshold() function. See this function's help file for more details.

rotate3D *Three-Dimensional Rotation Matrix*

Description

Render a 3-Dimensional projection matrix given positive or negative degree changes in yaw, pitch, and / or roll.

Usage

```
rotate3D(yaw, pitch, roll)
```

Arguments

`yaw`	z-axis change in degrees; look left (+) or right (-). Consider this a rotation on the x-y plane.
`pitch`	y-axis change in degrees; look up (-) or down (+). Consider this a rotation on the x-z plane.
`roll`	x-axis change in degrees; this change appears as if you touch head to shoulders: right roll (+) and left roll (-).

Details

When plotting with the package scatterplot3d, the default perpective is such that the pitch action appears as a roll while the roll action appears as a pitch.

This function is used only in data generation of the package vignette. This function is called by rotateScale3D().

Value

A 3 * 3 projection matrix of the degree changes entered.

`rotateScale3D` *Three-Dimensional Rotation and Scaling Matrix*

Description

Render a 3-Dimensional projection matrix given positive or negative degree changes in yaw, pitch, and / or roll and increment or decrement feature scales.

Usage

```
rotateScale3D(rot_angles = c(0, 0, 0), scale_factors = c(1, 1, 1))
```

Arguments

`rot_angles`	a list or vector containg the rotation angles in the order following: yaw, pitch, roll. Defaults to <0,0,0>.
`scale_factors`	a list or vector containing the values by which to multiply each dimension. Defaults to <1,1,1>.

Details

See the function commentary of "rotate_3D" for a brief explination of how these angles behave in scatterplot3d functionality (from package scatterplot3d).

This function is used only in data generation of the package vignette. This function calls rotate3D().

Value

A 3 * 3 projection matrix of the degree and scale changes entered.

`threshold` *Non-parametric Threshold Estimation*

Description

Calculate the non-parametric critical value estimates for the SPE and T2 monitoring test statistics.

Usage

```
threshold(pca_object, alpha = 0.001, ...)
```

Arguments

`pca_object`	A list with class "pca" from the internal pca() function
`alpha`	The upper 1 - alpha quantile of the SPE and T2 densities from the training data passed to this function. Defaults to 0.001.
`...`	Lazy dots for additional internal arguments

Details

This function takes in a pca object returned by the pca() function and a threshold level defaulting to 0.1 quantile is set this low to reduce false alarms, as described in Kazor et al (2016). The function then returns a calculated SPE threshold corresponding to the 1 - alpha critical value, a similar T2 threshold, and the projection and Lambda Inverse (1 / eigenvalues) matrices passed through from the pca() function call.

This internal function is called by faultFilter().

Value

A list with classes "threshold" and "pca" containing: SPE_threshold = the 1 - alpha quantile of the SPE density. T2_threshold = the 1 - alpha quantile of the Hotelling's T2 density. projectionMatrix = a projection matrix from the data feature space to the feature subspace which preserves some specified proportion of the energy of the data scatter matrix. This specified energy proportion is user specified through the var.amnt argument in the pca() function. LambdaInv = a diagonal matrix of the reciprocal eigenvalues of the data scatter matrix.

Index