DeviationAnomalyDetection operator

This operator searches for anomalies on the base of the standard-deviation. First input port: data, second input port: deviation information. The operator uses the deviation information from the DeviationLearn operator and compares the value of the tuple to the last learned deviation information without this tuple. If the value of the tuple is out of the normal range around the mean, the tuple is considered as an anomaly. The operator has some additional settings which can be used to find anomalies in some special situations.

Parameters

- **interval** Defines, how many standard deviations are allowed for a tuple to be different from the mean. 3.0 is the default value. Choose a smaller value to get more anomalies.
- **attribute** Name of the attribute which should be analysed
- **GROUP_BY** To group the tuples and learn a unique deviation for each group. If used with a deviationLearn operator, use the group attribute it produces as group_by in this operator.
- **fastGrouping** Use hash code instead of tuple compare to create group. Potentially unsafe!
- **windowChecking** If true, it's checked if the last (tumbling) window had an anomaly. If two following windows have at least one anomaly per window, all non-anomaly tuples between the two anomalies will be send, too. AnomalyScore for the non-anomaly tuples between the anomalies is Double.MINVALUE.
- **onlyFirstAnomaly** If you use windowChecking and set this option true, then an anomaly tuple is only send, if the last window had no anomaly. You get exalctly one tuple at the beginning of an anomal phase.
- **reportEndOfAnomalies** If you use windowChecking and set this option true, then a tuple with anomaly score = 0 is send, if the last window had an anomaly, but the current has not.
- **deliverUnlearnedElements** If you want to get tuples for which no deviation information is available (e.g. the first tuple), you can set this to true. Default is false.
- **tuplesToWait** If you want the operator to learn for a while before it starts to analyse, you can set that the operator has to wait x tuples (each group has its own counter) before it starts to analyse. Default is 0.
- **maxRelativeChange** If you want the operator to learn for a while before it starts to analyse, you can set that the operator has to wait until the relative change between the last mean and the new mean from the operator before this one is lower than the given value. This avoids early false-positives. Can be used together with 'tuplesToWait'. Default is 0.
- **timeSensitive** If you do an interval analysis and want to check if the duration since the last tuple when a punctuation arrives, you can use this option. Assumes, that the attribute which is analysed is the time between two tuples. Only sends a tuple, if the time between the last sent anomaly-tuple based on punctuations and the next anomaly-tuple based on punctuations is an anomaly itself. Does the check for all groups. Default is false.

Example

```
#PARSER PQL
#ADDQUERY
/// Use the online training mode to learn the mean and the standard deviation
deviationLearner = DEVIATIONLEARN({
                        trainingmode = 'ONLINE',
                        nameofparameter = 'temp'
                      },
                      System.manual
                    )

/// Compare the current tuple with the learned values
deviationAnalysis = DEVIATIONANOMALYDETECTION({
                        interval = 3.0,
                        attribute = 'temp'
                      },
                      1:deviationLearner,
                      0:deviationLearner
                    )
```

Extra settings

WindowChecking

The windowChecking option allows the operator, to use a window and check, if there is at least one anomaly in that window. If the data stream has anomal phases where some tuple may are "normal", these "normal" tuples can be marked as anomalies if they are between two windows which have both at least one anomaly. The figure below explains that behavior. The squares are tuples, each tuple is in exactly one window (a tumbling window is used). The tuples with a 0 are normal tuples, the tuples with a 10 are anomal tuples. With the option "windowChecking = true" the tuples between those two anomalies are also marked as anomalies and are therefore in the output stream.

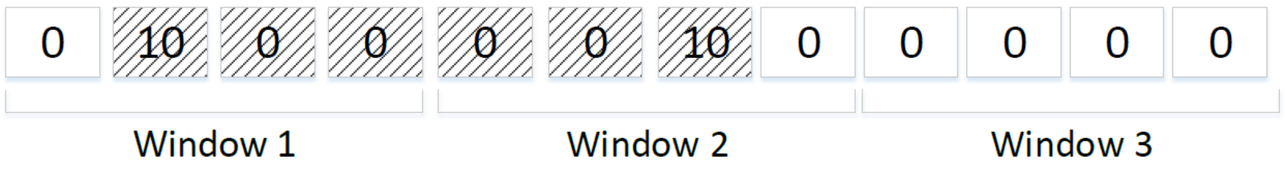

Figure 1: WindowChecking

Example

The code below shows an example with the windowChecking option. Here, some more special settings are used. The operator won't report the anomalies between the first and the last anomaly in an anomaly sequence. If you again take a look at the example figure above, the zeros between the "10"-tuples would not be reported with this options.

The DeviationLearn operator only learns the fist 100 tuples, because they are considered as normal.

```
#PARSER PQL
#RUNQUERY
/// A tumbling window -> We will search for anomalies in a window
windowOp = ELEMENTWINDOW({
                size = 20,
                advance = 20
              },
              System.fridgeVibration
            )

/// Learn how the "normal" area looks like
intervalLearn = DEVIATIONLEARN({
                    attribute = 'vibration',
                    trainingmode = 'TUPLE_BASED',
                    tuplestolearn = 100
                  },
                  windowOp
                )

/// Checks, if the current window has an anomaly
intervalDetect = DEVIATIONANOMALYDETECTION({
                      attribute = 'vibration',
                      interval = 3.0,
                      windowchecking = 'true',
                      onlyfirstanomaly = 'true',
                      reportendofanomalies = 'true'
                    },
                    1:intervalLearn,
                    0:intervalLearn
                  )
```