DeviationLearn operator

This operator learns the mean and the standard deviation of a single value and writes it into the output. On port 0, the learned values (mean and standard deviation) are put out, on port 1 the tuples which are used to learn the deviation. The operator puts out the original tuples with the groupId the operator chose for the tuple. This is necessary for the following operator to be able to map the deviation values from port 0 to the correct tuples on port 1.

Parameters

- **exactCalculation** If set to true, it uses exact calculation for window mode (recalc values every new tuple). This may be slower. Unexact calculation may be faster, but unaccurate, especially if the mean changes dramatically over time.
- **tuplesToLearn** The number of tuples that will be used to learn the operator if 'tupleBased' is the choosen trainingMode.
- **trainingMode** The training mode for this operator
- **mean** If you want to set the mean manually (with trainingMode = manual) you can do this here.
- **standardDeviation** If you want to set the standard deviation manually (with trainingMode = manual) you can do this here.
- **attribute** Name of the attribute which should be analysed
- **uniqueBackupId** A unique ID for this operator to save and read backup data.
- **GROUP_BY** To group the tuples and learn a unique deviation for each group
- **fastGrouping** Use hash code instead of tuple compare to create group. Potentially unsafe!

Example

In this example, the online-algorithm is used to learn the deviation. Therefore, no window is needed. The deviation is learned for the parameter "temp". The following operator uses the original tuples with the group of the DeviationLearn operator (1:deviationLearner) and the learned deviation (0: deviationLearner).

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

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

Backup

The operator can save the learned values info a database. The learned information is saved, not the tuples which learned the operator. When the query starts, the learned information is read from the database and written into the operator. This only happens once. The backup happens every time the operator updates the values. The PQL code belows shows, how to use this option.

```
#PARSER PQL
#DEFINE BACKUPSCHEMA [['sum1', 'Double'],['sum2', 'Double'],['sumWindowSqr', 'Double'],['sumWindow', 'Double'],
['m2', 'Double'],['mean', 'Double'],['backupId', 'String'],['k', 'Double'],['standardDeviation', 'Double'],
['n', 'Double'],['group', 'Double'],['backupId', 'String']]
#ADDQUERY
/// Read backup data from the database
backupMongo = MONGODBSOURCE({
                  database = 'odysseus',
                  port = 27017,
                  host = 'localhost',
                  collectionname = 'condition'
                }
              )
/// Convert backup data to tuples
backupTuple = KEYVALUETOTUPLE({
            schema=${BACKUPSCHEMA},
            TYPE = 'Backup',
            KEEPINPUT = 'false'
          },
          backupMongo
        )
/// Run DeviationLearn and use backup data, if it exists ("Recovery")
deviationLearner = DEVIATIONLEARN({
                        trainingmode = 'ONLINE',
                        attribute = 'vibration',
                        uniquebackupid = 'dev1'
                      },
                      System.fridgeVibration,
                      backupTuple
                    )
/// Convert backup data from port 2 to a key value object
keyValueOp = TUPLETOKEYVALUE({
                  type='KEYVALUEOBJECT'
                },
                2:deviationLearner
              )
/// Save the backup data to the database
mongoSink = MONGODBSINK({
                database = 'odysseus',
                port = 27017,
                host = 'localhost',
                collectionname = 'condition',
                batchsize = 1,
                deletebeforeinsert = 'true',
                deleteequalattribute = 'backupId'
              },
              keyValueOp
            )
```