

# Oracle Metrics Association Mining

Bern Winter School on Machine Learning  
Mürren, January 2020

ML Project Presentation

Marco Bassi  
March 16, 2020

# Project Objective

- Find patterns in Oracle database metrics using Association Mining.
- Result consists of association rules  $\{lhs\} \Rightarrow \{rhs\}$ , e.g.  
 $\{\text{Enqueue Requests Per Sec, User Transaction Per Sec}\} \Rightarrow \{\text{User Commits Per Sec}\}$
- Association Mining is a machine learning method for identifying associations among items. The method is widely-used for market basket analysis, and for detecting fraudulent credit card use  
 $\{\text{butter, jelly}\} \Rightarrow \{\text{bread}\}$   
 $\{\text{torch light}\} \Rightarrow \{\text{battery}\}$

## Benefits and Drawbacks

- 😊 unsupervised learning, i.e. no training and no labels
- 😊 very large number of variables, and large and complex data
- 😊 fast
- 😊 output (rules) is easy to understand
- 😞 not for small datasets
- 😞 takes effort to separate “obvious” rules from new insights
- 😞 beware of conclusions due to random patterns in the data

# Data Preparation – 1

- Input data selected from Oracle dynamic performance view DBA\_HIST\_SESSION\_METRIC
- Hourly snapshots of aggregated values for 158 metrics (avg, min, max, std.dev), for last 30 days:  
**Buffer Cache Hit Ratio, Memory Sorts Ratio, Redo Allocation Hit Ratio, User Transaction Per Sec, ...**
- Better: “realtime” metric values, taken every 60 secs; but unfortunately not available at the time of writing.
- Metrics values are numeric. “Itemize” values by subdividing into groups “high”, “medium” and “low” wrt. quantiles Q1 and Q3.
- For first attempt, only take “high” values into account.

# Analysis Procedure – 1

- Environment: R version 3.4.4, x86\_64 (Windows 10)
- Package arules version 1.6-3

**Step 1:** Load data into data structure itemMatrix (sparse matrix); see below 100 randomly selected samples



## Analysis Procedure – 2

**Step 2:** Use Apriori-Algorithm to find association rules

**Parameters:** threshold for *support*, *confidence* and minimum  $\{lhs\} + \{rhs\}$  size

- **support**: how frequently the itemset (items in  $\{lhs\}$  and  $\{rhs\}$  of the rule) occurs in the data samples.
- **confidence**: proportion of the samples, where the lhs itemset results into the rhs itemset.
- **minlen**: minimum total number of items in  $\{lhs\}$  and  $\{rhs\}$ .

**First attempt**

support=0.2, confidence=0.9, minlen=2  $\Rightarrow$  **set of 2176 rules**

# Analysis Procedure – 3

## Rules sorted by *lift*

```
[1]  {Physical Write Bytes Per Sec (2) H}          3.997175
      => {Physical Writes Per Sec (2) H}
[2]  {Physical Writes Per Sec (2) H}              3.997175
      => {Physical Write Bytes Per Sec (2) H}
[3]  {User Transaction Per Sec (2) H}            3.997175
      => {User Commits Per Sec (2) H}
[4]  {User Commits Per Sec (2) H}                3.997175
      => {User Transaction Per Sec (2) H}
[5]  {Host CPU Usage Per Sec (2) H}              3.997175
      => {Host CPU Utilization (%) (2) H}
...

```

**Lift:** a measure for the importance of a rule. The  $\{lhs\}$  and  $\{rhs\}$  itemsets are found together in samples more often by this factor, than would be expected by chance.

The rules shown above are of absolutely no use!

# Improving model performance – 1

## Subsetting the rules, e.g. for metrics concerning the Library Cache

```
lc.rules <- subset(rules, items %pin% "Library Cache") ⇒ 4 rules
```

```
[1] {Library Cache Miss Ratio (2) H} 3.661220  
    => {Hard Parse Count Per Txn (2) H}  
[2] {Hard Parse Count Per Txn (2) H} 3.661220  
    => {Library Cache Miss Ratio (2) H}  
[3] {Library Cache Miss Ratio (2) H,  
     Redo Writes Per Txn (2) H} 3.893877  
    => {Enqueue Waits Per Txn (2) H}  
[4] {Enqueue Waits Per Txn (2) H,  
     Library Cache Miss Ratio (2) H} 3.828979  
    => {Redo Writes Per Txn (2) H}
```

Rules [3] resp. [4] are worth further exploring



## Improving model performance – 2

Creating rules more specifically, e.g. for {rhs} concerning the Library Cache:

```
lc_rules <- apriori(data=im,  
  parameter=list(support=0.1, confidence=0.9, minlen=2, maxlen=3),  
  appearance=list(default="lhs",  
    rhs=c("Library Cache Hit Ratio (2) H",  
      "Library Cache Miss Ratio (2) H"))) ⇒ 144 rules
```

Subsetting for “Library Cache Miss Ratio (2) H” on the {rhs} shows some not obvious associations, e.g.

```
{Enqueue Waits Per Txn (2) H, 3.826479  
  Redo Allocation Hit Ratio (2) H}  
=> {Library Cache Miss Ratio (2) H}
```

However, it might as well just be a random pattern; needs some more exploration with a better dataset.

## Additions & Conclusions

- Association rules resulting from the Apriori algorithm are highly redundant. However, there are techniques to remove the redundancies.
- Extracting rules that are actionable requires an extra effort.
- In setting the parameters for the Apriori algorithm, You have to trade of *support* against *confidence*:
  - A high support threshold might exclude rare itemsets which might result in high confidence rules
  - A low support threshold will produce a huge number of rules
- An extension to association mining with itemsets (market baskets), is mining for frequent sequential patterns; e.g. for DNA fingerprinting.