RuleVisualization 1.0 - User guide

Contents

1 Installation

1.1 Running application

Application consists of client and server folder. Requirements:

- Java 8 or higher (`java -version`),
- Internet browser.

To run application:

1. Run server on specified port number (8081 by default):
   `java -jar ./server/server.jar <port_number>`

2. Run client by opening `index.html` file in browser.

3. Enter address of server URL on `index.html` webpage.

1.2 Application development

Server development (Java):

1. Compile project.
   `mvn compile`

2. Modify code.

3. Generate JAR file into `target/server.jar`.
   `mvn clean install`

Client development (Vue.js):

1. Install modules defined in `package.json` (first project run):
   `npm install`

2. Enable hot reloading on `http://localhost:8080`:
   `npm run serve`

3. Modify code.

4. Build application into `dist` folder:
   `npm run build`

2 Setup

Setup tab is used for loading data and configuring attributes and characteristics. In general 4 types of objects can be distinguished: attributes, rules, examples and characteristics (rules file).

2.1 Loading data

Data can be entered from local filesystem:

1. Enter URL for server with ruleLearn library (see running server: 1.1).

2. Browse file with attributes (JSON format) - required.

3. Browse file with rules (XML format) - required.

4. Browse file with examples (JSON format) - optional.

5. Click Load Button.

6. After loading data: status message, tabs and configuration will be displayed (or updated).

Alternatively data can be loaded from server demo datasets:

1. Enter URL for server with ruleLearn library (see running server: 1.1).

2. Select dataset.

3. Click Load demo.

4. After loading data: status message, tabs and configuration will be displayed (or updated).

Format of files is compatible with format of files generated by ruleLearn library (github.com/ruleLearn/rulelearn). In order to get familiar with format, there is option for downloading sample (demo) files by clicking "Download demo".

2.2 Configuration - optional

Configuration enables user to explore loaded attributes and characteristics. This step is fully optional it can be omitted. Name of this objects can be displayed with another name (e.g. shorter name) in filtering and characteristics section. Type, value-type, domain, active are object properties loaded from JSON file (read only). Property Filter is used for show/hide objects in filtering section, Table - show/hide in Rules table. Min and max are the minimum and maximum values of objects' domains. They are used for: filtering section (slider range), **computing similarity and coloring edges/nodes in rules graph (based on domains' ranges)**. Steps are only used for filtering section (how many intervals slider contains). Values Min and Max are by default:

- for attributes: read from rules file (min and max occurence),
- for characteristics: predefined values if ranges are known for characteristic (e.g. Strength is from 0 to 1), otherwise read from rules file (e.g. Support is from 0 to max occurence in rules file).

Attributes	Characteristics							
Display name	**Type**	**Value type / domain**	**Active**	**Filter**	**Min**	**Max**	**Steps**
AttackSourceReputationSc	gain	real	✔	✔	0.75	0.93	20
ID	id	uuid	✔	☐			
SeverityForAttackCategory	gain	NA, low, med, high, critical	✔	✔			

Attributes	Characteristics					
Display name	**Active**	**Table**	**Filter**	**Min**	**Max**	**Steps**
AConfirmation	✔	✔	✔	0	1	20
C1Confirmation	✔	✔	✔	0	1	20

3 Filtering and matching rules

Application stores in runtime: source ruleset (rules read from file), and context ruleset (source ruleset after applying filtering and matching). Filtering and matching can be used to change context rules. Context rules are always the same rules for all sections of

application. It means that applying filter in one section (e.g. rules list), results in all other sections (graph, attributes matrix, statistics). Matching rules to objects can be interpreted as applying a filter.

Interpretation for filters selected on above figure:

> Select rules with SConfirmation between 0.209 and 0.996, Strength between 0.001 and 0.792, Support between 1 and 458. Every rule has to have condition C1 (with attribute SourceAssetCriticality, operator $\geq$, value between "very low" and "high") **or** decision C2 (with attribute Priority, operator $\leq$ and value between 2 and 5). This rule mustn't contain condition C3 (with attribute TargetAssetCriticality and value between "medium" and "high".). Finally it has to also cover object "bec78441".

Applying filters consists of:

- setting lower and upper limits of every characteristic that can occure in rule (applies after clicking Apply),

- setting occurence (INCLUDE - tick, EXCLUDE - minus, IGNORE - blank), operator ($\leq, \geq, \in$) and lower/upper limits of every attribute (applies after clicking Apply),

- selecting object, (covering filter applies after clicking Match button).

Mechanism of filtering insert rules from source ruleset into context ruleset. Every context rule:

1. "contains" at least one "ticked" attribute if JoinOperator = or, or, "contains" all "ticked" attributes if JoinOperator = and,

2. not "contains" attributes with minus sign,

3. may contain or may not contain blank attributes (without tick and minus) - these attributes are ignored and don't reduce context ruleset (operator and limits for this attributes are also ignored).

"Contains" means that, in rule, exists condition or decision with: chosen attribute, chosen operator and with value that is between limits. For instance when operator is equal $\geq$ and limits is $[2, 5]$, it will match conditions/decisions: $\geq 2, \geq 3, \geq 4, \geq 5$.
Operator $\in$ means: ANY operator. "Reset" button removes filter/match. If filtering or matching reduce source ruleset, then tab is red-colored (as on image).

3.1 Match warning

Matching examples (selecting covering rules) is performed with use of Rule.covers method from rulelearn library. Examples can be edited by user. After modification, example is marked as "dirty". Clicking Match button sends all "dirty" objects to server in order to match rules again with ruleLearn library. **In order to refresh coverage similarity computations in rules graph, user is required for clicking "Recompute" in Graph tab**.

4 Rules list

This section represents list of context rules. After resetting filters and object matching, list displays all loaded decision rules. List has 3 features:

1. Sorting rules by characteristic value or length of condition part (by clicking on characteristic/conditions header of column),

2. Enable/disable wrap of condition part content,

3. Filtering covered examples by rule by clicking on rule's condition part (covered examples are displayed in Match section).

5 Statistics

Statistics tab is used for displaying statistics of characteristics and attributes occurring in context ruleset. When user clicks on attribute or characteristic, it will display histogram of characteristic/attributes values occurring in context ruleset.

Top attributes can be sorted by count or "importance" measure. "Importance" is a sum of attribute's occurrences in conditions multiplied by rule strength. This sum is divided by average rule strength in dataset.
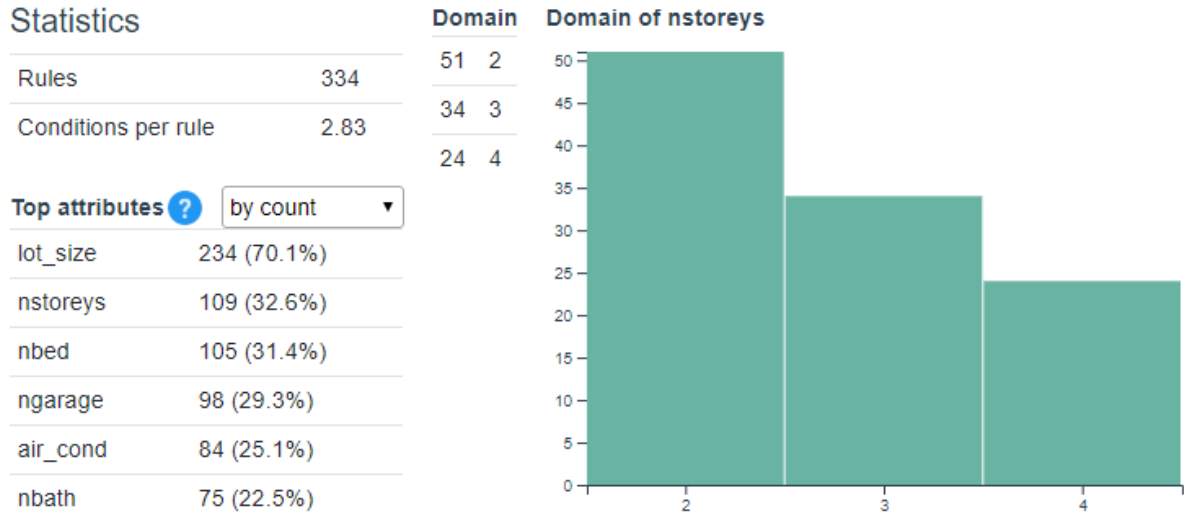
6 Attributes matrix

Matrix of attributes is used for showing correlations between attributes. Value of matrix's element represents coocurrence of attributes a_i and a_j. It is number of coocurrences divided by: top attribute occurences or row attribute occurences (see Cell value option). This tab has 4 options to configure:

1. Contrast - every value of matrix element is raised to the power of value (contrast/10),

2. Matrix size - enables to control visual size of matrix,

3. Cell value - coocurences count is divided by top attribute's coocurrences (Absolute) or by row attribute's occurrences (Relative to row),

4. Cell size - Size of element is fixed (Fixed) or proportional to occurrences of attributes a_i and a_j (Weighted).

7 Rules graph

Rules graph is used for visualizing decision rules. Nodes represent characteristics selected by user. Edges represent similarity of rules pair. There are 2 types of similarity:

- coverage - similarity of rules' coverage, number of objects covered by both rules divided by number of objects covered by at least one of this 2 rules, **do not confuse with rule's Coverage characteristic**,

- semantic - similarity based on similarity of syntax of rules' conditions parts.

Semantic similarity is computed by algorithm presented below:

Algorithm 2: Similarity of rules pair

Input: ruleA, ruleB

       attr(r) - set of attributes occuring in conditions part of rule r

       v(r, a) - characteristic value for condition with attribute a in rule r

       len(a) - length of domain of attribute a

Output: similarity(ruleA, ruleB)

similarity $\leftarrow$ 0;

A $\leftarrow$ attr(ruleA);

B $\leftarrow$ attr(ruleB);

foreach $a \in (A \cap B)$ **do**

    attrSimilarity $\leftarrow 1 - \alpha \cdot \dfrac{|v(\text{ruleA}, a) - v(\text{ruleB}, a)|}{len(a)}$;

    similarity $\leftarrow$ similarity $+$ attrSimilarity;

end

similarity $\leftarrow \dfrac{2 \cdot \text{similarity}}{|A| + |B|}$;

There are a lot of options for customizing rules graph:

1. Rules - show graph generated for rules with decision "at least" or with decision "at most",

2. Operator - edge between nodes (rules) will exists if one of threshold (Semantic, Coverage) will be exceeded (operator = or), or if both thresholds will be exceeded (operator = and),

3. Semantic threshold - minimum value of semantic similarity (see Operator option),

4. Coverage threshold - minimum value of coverage similarity (see Operator option), **do not confuse with rule's Coverage characteristic**,

5. Node size - size of node is proportional to selected characteristic,

6. Node color - color of node is proportional to selected characteristic (from white, through yellow, red, to black),

7. Edge size - width of edge is based on selected similarity measure,

8. Edge color - color of edge is based on selected similarity measure (from yellow through red to black),

9. Graph mode - changes behavior when user clicks on node.

10. Refresh - Rerun simulation (e.g. after applying filter, changing options: Thresholds, Operator, Rules, Physics),

11. Recompute - this option should be used only for recomputing Coverage similarity after changing examples set (read more: 3.1).

Physics options are options defined for forceSimulation in D3. You can also find their meaning in books about D3.

1. Center - strength of force that attracts nodes to the center of graph,

2. Collide - strength of force that repels nodes from each other,

3. Collide radius - maximum operating range of Collide force,

4. Link - strength of force that attract nodes connected with edge,

5. Link distance - space between nodes,

6. Charge - strength of force that repels nodes from each other (D3 forceManyBody),

7. Charge theta - higher theta increases speed, decreases accuracy of simulation,

8. Iterations - Number of simulation's ticks (steps, iterations).

Similarity

Rules	⦿at least ⭕at most	☑ Physics	
Operator	⭕or ⦿and	Center	0.1
Semantic	⭕——— 0	Collide	0.33
Coverage	——⭕— 0.33	Collide radius	100
Node size	fixed ▾	Link	1
Node color	fixed ▾	Link distance	25
Edge size	coverage ▾	Charge	-50
Edge color	semantic ▾	Charge theta	1
Graph mode	⭕Drag ⦿Read	Iterations	300

Refresh Recompute ❓