Rules mining algorithm

Oleg Shpynov
os@jetbrains.com

April 13, 2016

Mining space

Predicates mining is an optimization problem: given formula $t \in F$ find formula $c \Big| \forall c' \in F \; metrics(c \rightarrow t) \geq metrics(c' \rightarrow t)$. The size of the search space for N predicates is $>> 2^N$ since all possible formulas should be counted.

This **exponential** growth in the number of features or predicates causes both computational problems and statistical problems (overfitting).

Approaches

- Greedy
- Forward feature selection
- Backward feature selection

Backward approaches are not applicable, since the set of all the available formulas doesn’t have maximal element.

Long story short

- Started with CNF formulas enumeration - slow.
- Created greedy algorithm with injections - too greedy, counter examples are easy to build.
- Dynamic programming algorithm - reasonable fast.
- Counter examples lookup: take real world predicates. The problem is in regularization [1].
- Automatic counter examples building - failed.
- Manual counter example - success.

[1] https://github.com/JetBrains-Research/epigenome/issues/649

Helpers

procedure selectTop(conditions, target, topN)

return *topN* best rules of $\{c \rightarrow target | c \in conditions\}$ by metrics

end procedure

procedure inject(f, a)

return set of formulas constructed by applying $\{AND, OR, NOT, REPLACE\}$ operations to any subformula of f and atomic predicate a

end procedure

procedure inject(formulas, atomics)

return $\cup\{inject(f, a) | \forall f \in formulas, \forall a \in atomics\}$

end procedure

Inject example

```
    fun testInjectNotOr() {
        val predicates = PredicatesInjector.injectPredicate(p("0 OR NOT 1"), p("2"))
        assertEquals("""(0 OR NOT 1) AND 2
(0 OR NOT 1) AND NOT 2
0 AND 2 OR NOT 1
0 AND NOT 2 OR NOT 1
0 OR 2
0 OR 2 AND NOT 1
0 OR 2 OR NOT 1
0 OR NOT (1 AND 2)
0 OR NOT (1 AND NOT 2)
0 OR NOT (1 OR 2)
0 OR NOT (1 OR NOT 2)
0 OR NOT 1 AND NOT 2
0 OR NOT 1 OR NOT 2
0 OR NOT 2
2
2 OR NOT 1
NOT 1 OR NOT 2
NOT 2""",
                ts(predicates))
    }
```

Dynamic programming

Invariant:

$best[i]$ stores *top* conditions built from $predicates[j]$, where $j \leq i$, i.e. all the complex formulas in $best[i]$ consist of atomic predicates with smaller indexes and **contain** atomic predicate number i.

Failure:

Given atomic predicates P, target t and metric function, algorithm **fails** to find solution c if $\forall i \big| P[i] \in c$ formula $c \neq inject(best[j], P[i])$.

Algorithm

procedure optimize(predicates, target, topN)
    $best \leftarrow []$
    for i in $0 \ldots len(predicates) - 1$ **do**
        $atomics \leftarrow \{predicates[i], \neg predicates[i]\}$
        if $i = 0$ **then**
            $formulasToTest \leftarrow atomics$
        else
            $formulasToTest \leftarrow \sum_{j=0}^{i-1} inject(best[j], atomics)$
        end if
        $best[i] \leftarrow selectTop(formulasToTest, target, topN)$
    end for
    return $selectTop(\sum_{i=0}^{len(best)-1} best[i], target, topN)$
end procedure

Questions

Questions:

- Correctness?
- Order dependent?
- Counter example?

Counter example lookup

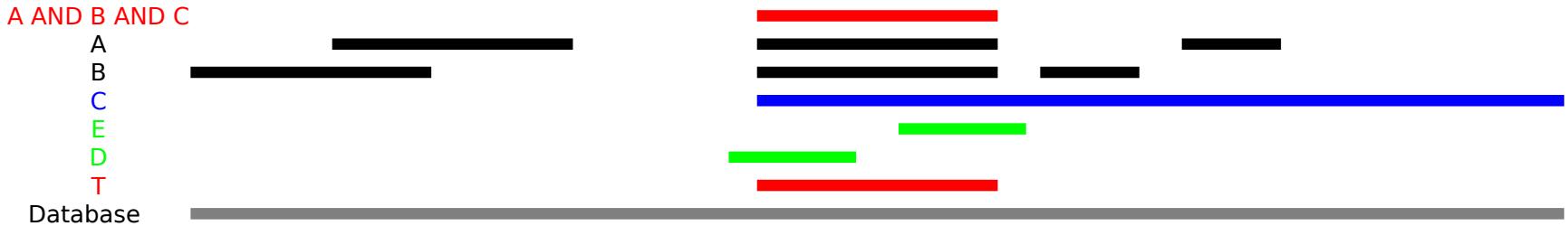
Construct the smallest possible counter example.

- Data size 100, 1000, 10000
- For i in $1 \ldots 9$ generate 10x **random** predicates with $p = \frac{i}{10}$
- Optimize top rule with *top* parameter = 1, 10, 20
- Optimize on direct, **reversed** and **random** order of atomic predicates
- If rule with different top convictions found:
  Print predicates and rules. Exit.
- Try harder!

Result: no counter examples found.

Counter example

Build 3 sets, so that A AND B AND C is solution, but A AND B is neither in $best[A]$ nor in $best[B]$.



Consider the following order: D, E, A, B, C for target T.

$best[D]$ = D

$best[E]$ = E

$best[A]$ = A AND D

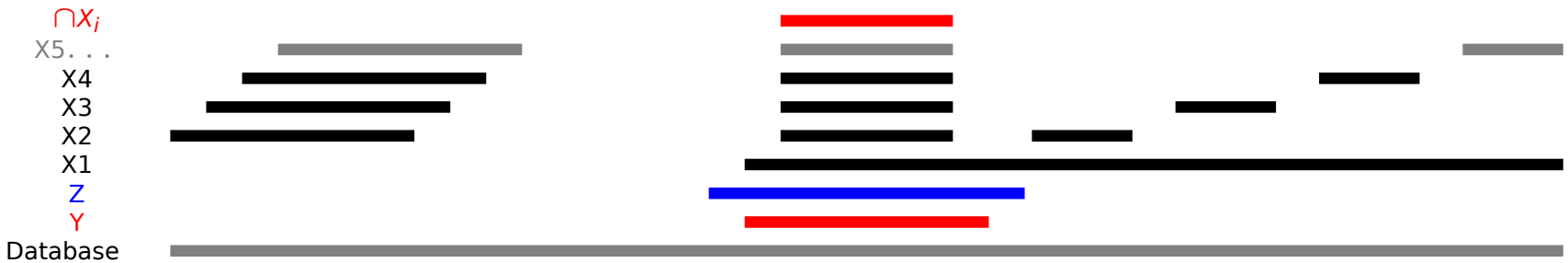
$best[B]$ = B AND E

$best[C]$ = C AND D

Counter example

Counter example set $X = \{x_i\}$, target Y and Z |

- $Z \setminus Y$ is *"small"*
- $\forall i, j x_i \cap x_j \neq \emptyset$ and $x_i \cap x_j \not\subseteq Y$
- $\cap x_i \neq \emptyset$ and $\cap x_i \subseteq Y$.



Problem with order $Z, X_1, X_2, \ldots$: $best[X_i] = Z$ AND X_i.

Once Z is not before $X_1, X_2, \ldots$, algorithm succeeds [2].

Solution: Order predicates by *"distance"* descending, i.e. start search with the most distant predicates.

[2] regularization can still be an issue

Counter example experiment

Distance: $D(p, target) = \#(p \setminus target)$.
Sort predicates | $\forall i < j \; D(p_i, \; target) \geq D(p_j, target)$.
Avoid predicates *"close"* to the target on initial states of dynamic programming algorithm[3].
Experiment:

- 2 scores: conviction with and without regularization
- Randomized order
- Distance order
- Simple greedy algorithm with correct order

[3] Possible we can prove correctness, since $best[i]$ does not increase distance

RuleGeneratorCounterExample.kt[4]

```
T = [-100;100)
A = [-1000;-800) OR [-100;100) OR [100;200)
B = [-100;100) OR [-900;-700) OR [200;300)
C = [-100;1000)
D = [-120;-50)
E = [50;120)

Standard score function
Best rule [conviction=180.010 score=164.010 cts=200,200,200] A AND B AND C => T
Rule correct order [conviction=45.002 score=37.002 cts=50,200,50] A AND D => T
Ordered...
Ordered attempts: 200/200 Min top: 3 Max top: 3 Avg top: 3.0
Ordered: done in 5.558 s
Shuffled...
Shuffled attempts: 200/200 Min top: 1 Max top: 4 Avg top: 2.355
Shuffled: done in 3.406 s
Greedy ordered...
Greedy Ordered attempts: 200/200 Min top: 5 Max top: 5 Avg top: 5.0
Greedy ordered: done in 5.480 s

Score function = conviction wo regularization
Best rule [conviction=180.010 score=180.010 cts=200,200,200] A AND B AND C => T
Rule correct order [conviction=180.010 score=180.010 cts=200,200,200] A AND B AND C => T
Ordered...
Ordered attempts: 200/200 Min top: 1 Max top: 1 Avg top: 1.0
Ordered: done in 974.4 ms
Shuffled...
Shuffled attempts: 200/200 Min top: 1 Max top: 6 Avg top: 2.27
Shuffled: done in 4.941 s
Greedy ordered...
Greedy Ordered attempts: 200/200 Min top: 1 Max top: 1 Avg top: 1.0
Greedy ordered: done in 510.6 ms
```

[4] https://github.com/JetBrains-Research/epigenome/commit/82b5832c65310643ec1ea91f095c13e8d25c409e

Summary

Questions:

- Correctness? - NO
- Order dependent? - YES

Workaround:

- Almost impossible to build counter example automatically
- Even in complicated cases order matters
- Use regularization
- Shuffle predicates order
- Start with *"correct"* order
- TODO: generate counter examples for given number of X_i and see the dynamics.

More counter examples

```
Standard score function and predicates: 6
X0 = [-100;1000)
X1 = [-10000;-9000) OR [-120;100) OR [100;200)
X2 = [-120;100) OR [-9900;-8900) OR [200;300)
X3 = [-120;100) OR [-9800;-8800) OR [300;400)
X4 = [-120;100) OR [-9700;-8700) OR [400;500)
X5 = [-120;100) OR [-9600;-8600) OR [500;600)
Z = [-120;120)
Best rule [conviction=198.000 score=70.000 cts=200,200,200] X0 AND X1 AND X2 AND X3 AND X4 AND X5 => T
Rule correct order [conviction=5.795  score=3.795 cts=240,200,200] Z => T
Ordered...
Ordered attempts: 100/100 Min top: 3 Max top: 3 Avg top: 3.0
Ordered: done in 59.06 s
Shuffled...
Shuffled attempts: 100/100 Min top: 1 Max top: 3 Avg top: 1.76
Shuffled: done in 28.55 s
Greedy ordered...
Greedy Ordered attempts: 100/100 Min top: 11 Max top: 11 Avg top: 11.0
Greedy ordered: done in 3.619 min
```

7 and more predicates have less score than $Z => T$.

Even for 6 predicates $topN = 3$ is enough!

Shuffle + dynamic programming rulezzz!