Iterations for Propensity Score Matching in MonetDB

Michael Böhlen, **Oksana Dolmatova**, Michael Krauthammer, Alphonse Mariyagnanaseelan, Jonathan Stahl, Timo Surbeck

August 2020, ADBIS

Data

- Relation r:
  - Key: PatientID ID
  - Covariates:
    - Age A
    - Blood pressure P
    - Weight W
  - Treatment T $(1/0 = \text{yes/no})$
  - Outcome O $(1/0 = \text{recovered/sick})$

r

	ID	A	P	W	T	O
r_1	1	67	125	65	0	1
r_2	2	69	58	54	0	0
r_3	3	57	45	75	0	0
r_4	4	45	55	94	1	1
r_5	5	78	110	68	1	0
r_6	6	90	80	61	1	0

- All patients were sick before the treatment. Attribute O is the state after the treatment.

- **Task:** Form comparable cohorts of patients (to assess the effectiveness of the treatment).

Problem

Patients are often not comparable with each other:

r

ID	A	P	W	T	O
1	67	125	65	0	1
2	69	58	54	0	0
3	57	45	75	0	0
4	45	55	94	1	1
5	78	110	68	1	0
6	90	80	61	1	0

r'

ID	T	O
1	0	1
2	0	0
3	0	0
4	1	1
5	1	0
6	1	0

r"

ID	A	T	O
4	45	1	1
3	57	0	0
1	67	0	1
2	69	0	0
5	78	1	0
6	90	1	0

For different cohorts we get different conclusions:

- Conclusion for cohort with all (r'): treatment is not effective
- Conclusion for cohort with young (r''): treatment is effective

To get meaningful conclusions we build cohorts with comparable patients.

Propensity score [Rosenbaum and Ruben, 1983]

- The **propensity score** is the probability that a patient gets treated given her/his covariates:

$$\text{propensity\_score}(\mathbf{ID}_i) = P(\mathbf{T} = 1 | \mathbf{A}_i, \mathbf{P}_i, \mathbf{W}_i)$$

- Patients with the similar propensity scores are similar and we use the propensity score to build **cohorts of comparable patients**.

Background

- We work with relations and the **relational matrix algebra** (RMA[1]).
- RMA extends the relational algebra with matrix operations defined over relations:
  - $\bowtie$, σ, π, ...
  - `inv`, `mmu`, `add`, `tra`, ...
- SQL examples for relations $r(A, B, C)$ and $s(D, E)$:
  - `inv` `SELECT * FROM INV(r BY A);`
  - `mmu` `SELECT * FROM MMU(r BY A, s BY D);`
- For each relation the ordering of the rows is specified.

[1] O. Dolmatova, N. Augsten, and M. Böhlen, *A relational matrix algebra and its implementation in a column store*, SIGMOD, 2020

Propensity score estimation [Guo and Fraser, 2010]

- To compute the propensity score we must solve a logistic regression ($r1 =$ covariates, $r2 =$ treatment):

r1

ID	A	P	W
1	67	125	65
2	69	58	54
3	57	45	75
4	45	55	94
5	78	110	68
6	90	80	61

r2

ID	T
1	0
2	0
3	0
4	1
5	1
6	1

$$sigmoid(67 * X_A + 125 * X_P + 65 * X_W) = 0$$
$$sigmoid(69 * X_A + 58 * X_P + 54 * X_W) = 0$$
$$\ldots$$

- Coefficients $x = (X_A, X_P, X_W)$ are the solution of $sigmoid(r1 * x) = r2$
- The *sigmoid* function normalizes values to [0:1].
- The equation is overdetermined
  - The solution is iterative.
  - x is approximate.
- $sigmoid(r1 * x)$ is the **estimated propensity score**.

Iterative methods

- The coefficients are computed with an **iterative method**, e.g., gradient descent.
- The key properties of iterative methods:
  - The initial coefficients are often random.
  - The solution x is refined in each step of the iteration.
  - The iteration stops when the estimated values (e) are close to the target values ($r2$).
  - The size of x is fixed.

r1

ID	A	P	W
1	67	125	65
2	69	58	54
3	57	45	75
4	45	55	94
5	78	110	68
6	90	80	61

r2

ID	T
1	0
2	0
3	0
4	1
5	1
6	1

x

C	iT
A	-0.06
P	0.03
W	-0.02

First two steps of gradient descent over $r1$ and $r2$

Step 1:

x

C	iT
A	-0.06
P	0.03
W	-0.02

→ x

C	iT
A	-0.03
P	0.07
W	0.02

e

ID	eT
1	1
2	0.9
3	0.9
4	0.9
5	1
6	0.9

r2

ID	T
1	0
2	0
3	0
4	1
5	1
6	1

Step 2:

x

C	iT
A	-0.03
P	0.07
W	0.02

→ x

C	iT
A	-0.06
P	0.03
W	-0.01

e

ID	eT
1	0.2
2	0.8
3	0.8
4	1
5	0.7
6	0.9

r2

ID	T
1	0
2	0
3	0
4	1
5	1
6	1

SQL extension for iterative methods

To integrate iterative methods into SQL we extend the WITH clause:

```
WITH
  ITERATED r AS
    INITIAL (S1)
    REFINE  (S2)
    UNTIL   (P)
SELECT * FROM r;
```

- *r* is an iteratively refined relation of fixed size.
- S1 is a statement that initializes *r*.
- S2 is a statement that computes the refined values for *r* .
- P is a predicate to terminate the iteration.

Iteration example

- The following example SQL statement iteratively computes relation *s*:

```
WITH
  ITERATED s AS
    INITIAL ( SELECT ID, random[20:90] AS A,
                          random[30:150] AS P,
                          random[50:200] AS W
              FROM r )
    REFINE  ( SELECT ID, A*0.1, P*0.1, W*0.1 FROM s )
    UNTIL   ( ( SELECT MAX(W) FROM s ) < 1 )
SELECT * FROM s;
```

- *s* is initialized with *ID* from relation *r* and random values as covariates.
- Multiplies age, blood pressure, and weight in *s* by 0.1 in each step.
- Stops when maximal weight is smaller than 1.

This is a **shape preserving iteration**.

Contributions

We define **shape preserving iterations** that iterate over an iteratively refined relation of a fixed size.

- We offer *random initialization* that initializes iteratively refined relations with contextual information.
  - We prove that given input relations of sizes $m \times n$ and $l \times k$, RMA expressions can initialize relations of all necessary sizes $u \times v$, where $u, v \in \{m, n, k, l\}$.
- We define *stable queries* that refine values in iteratively refined relations and preserve their sizes.
- We offer efficient implementation of shape preserving iterations in MonetDB that allocates new memory only in the first step of an iteration and then reuses it.

Shape preserving iteration for gradient descent

A shape preserving iteration that performs gradient descent to compute the coefficients:

- $r1(ID, A, P, W)$ and $r2(ID, T)$ are input relations.
- $x(C, iT)$ is an iteratively refined relation.
- x is initialized with random numbers in iT and names of covariates in C.
- Values in x are refined in each step.
- x with refined coefficients is the result relation.

r1

ID	A	P	W
1	67	125	65
2	69	58	54
3	57	45	75
4	45	55	94
5	78	110	68
6	90	80	61

r2

ID	T
1	0
2	0
3	0
4	1
5	1
6	1

x

C	iT
A	-0.06
P	0.03
W	-0.02

$\rightarrow$

x

C	iT
A	-0.03
P	0.07
W	0.02

$\rightarrow \ldots \rightarrow$

x

C	iT
A	2.83
P	0.12
W	3.88

2.83 is how the age impacts the treatment.

Refinement step for gradient descent

The refine statement (S2) includes the following steps:

- The estimation of a treatment:
  $e = sigmod(r1 * x)$

```
SELECT ID, 1/(1+1/EXP(iT)) AS eT
FROM MMU ( r1 BY ID, x BY C );
```

- The difference between estimated and real treatment: $t1 = e - r2$

```
SELECT r2.ID, e.eT - r2.T AS eT
FROM r2 NATURAL JOIN e;
```

- The normalized difference:
  $d = t1/t1.length$

```
SELECT t1.ID, t1.eT/t2.N AS iT
FROM t1, ( SELECT COUNT(*) AS N FROM t1 ) AS t2;
```

- The gradient values:
  $g = r1^t * d = CPD(r1, d)$

```
SELECT * FROM CPD[C]( r1 BY ID,  d BY ID );
```

- The refinement of the values:
  $x = x - \alpha * g$

```
REFINE ( SELECT x.C, x.iT - α*g.iT AS iT
         FROM x NATURAL JOIN g );
```

Estimating propensity score

- Once coefficients are computed we can estimate the propensity score.
- Relational matrix multiplication between the covariates and the coefficients estimates the propensity score.

```
1   SELECT ID, 1/(1+1/EXP(iT)) AS PrSc
2   FROM MMU (  r1 BY ID, x BY C );
```

r1

ID	A	P	W
1	67	125	65
2	69	58	54
3	57	45	75
4	45	55	94
5	78	110	68
6	90	80	61

x

C	iT
A	2.83
P	0.12
W	3.88

p

ID	PrSc
1	0.438
2	0.241
3	0.690
4	0.732
5	0.421
6	0.944

p ordered by PrSc

ID	PrSc	
2	0.241	cohort 1
5	0.421	cohort 1
1	0.438	cohort 1
3	0.690	cohort 2
4	0.732	cohort 2
6	0.944	cohort 2

- With propensity score we can form the cohorts.

Conclusion

- We define shape preserving iterations to integrate iterative methods into the relational model.
- We implement shape preserving iterations in MonetDB to integrate iterative methods into databases.
    - We use iterative methods to compute logistic regression, k-means, linear approximation of matrix equations.
- Our implementation is efficient and leverages characteristics of iterative methods.

Thank you!