Privacy Proofs for OpenDP: Is_Equal Transformation

Grace Tian

Summer 2021

Contents

1 Algorithm Implementation

1.1 Code in Rust

The current OpenDP library contains the `make_is_equal` function implementing the is_equal function. This is defined in lines 62-71 of the file `manipulation.rs` in the Git repository (`https://github.com/opendp/opendp/blob/21-impute/rust/opendp/src/trans/manipulation.rs#L62-L71`).

1.2 Pseudo Code in Python

Preconditions

To ensure the correctness of the output, we require the following preconditions:

- **User-specified types:**
  - Variable `value` must be of type `TI`
  - Type `T` must have trait `PartialEq`

Postconditions

- Either a valid `Transformation` is returned or an error is returned.

```
def make_is_equal(value : TI):
    input_domain = (VectorDomain(AllDomain(TI));
    output_domain = (VectorDomain(AllDomain(bool));
    input_metric = SymmetricDistance();
    output_metric = SymmetricDistance();


    def Relation(d_in: u32, d_out: u32) -> bool:
        return d_out >= d_in*1

    def function(data: Vec(TI)) -> Vec(Bool):
        return list(map(data == value))

    return Transformation(input_domain, output_domain, function, input_metric, output_metric
    , Relation)
```

(grace) For the next round of the updates, will need to change pseudocode so that it returns the result of a make row by row transformation (which the code does).

1.3 Proof

Theorem 1.1. *For every setting of the input parameters* `value` *to* `make_is_equal` *such that the given preconditions hold, the transformation returned by* `make_is_equal` *has the following properties:*

1. (Appropriate output domain). *If vector v is in the* `input_domain`*, then* `function(v)` *is in the* `output_domain`*.*

2. (Domain-Metric Compatibility). *The domain* `input_domain` *matches one of the possible domains listed in the definition of* `input_metric`*, and likewise* `output_domain` *matches one of the possible domains listed in the definition of* `output_metric`*.*

3. (Stability Guarantee). *For every pair of elements v, w in* `input_domain` *and for every pair (*`d_in`*,* `d_out`*), where* `d_in` *is of the associated type for* `input_metric` *and* `d_out` *is the associated type for* `output_metric`*, if v, w are d_{in}-close under* `input_metric` *and* `Relation`*(*`d_in`*,* `d_out`*) =* `True`*, then* `function`*(v),* `function`*(w) are d_{out}-close under* `output_metric`*.*

Proof.

1. **(Appropriate output domain).** In the case of `make_is_equal`, this corresponds to showing that for every vector v of elements of type `TI`, `function`(v) is a vector of elements of type `bool`.

   The `function`(v) has type `Vec(TI)` follows from the assumption that element v is in `input_domain` and from the type signature of `function` in line 11 of the pseudocode (Section 1.2), which takes in an element of type `Vec(TI)` and returns an element of type `Vec(Bool)`. If the Rust code compiles correctly, then the type correctness follows from the definition of the type signature enforced by Rust. Otherwise, the code raises an exception for incorrect input type.

2. **(Domain-metric compatibility).**

   Symmetric distance is compatible with `VectorDomain(AllDomain(TI))` for any generic type `TI`, as stated in "List of definitions used in the pseudocode". The theorem holds because for `make_is_equal`, the input domain is `VectorDomain(AllDomain(TI))` for generic type `TI` and the output domain is `VectorDomain(AllDomain(bool))`.

3. **(Stability guarantee).** Because `Relation`$($`d_in`$,$`d_out`$) =$ `True`, it follows that `d_in` $\leq$ `d_out` by the `is_equal` stability relation defined in the pseduocode.

   Since vector inputs v, w are `d_in`-close, then the symmetric distance is bounded by `d_in` by definition the symmetric distance is bounded by d_{in}: $d_{Sym}(v, w) \leq \texttt{d\_in}$.

   We apply the histogram notation, as stated in "List of definitions used in the pseudocode", to rewrite the symmetric distance in terms of elements z in `function`(v) and `function`(w)

   $$d_{Sym}(v, w) = \|h_v - h_w\|_1 = \sum_z |h_v(z) - h_w(z)|.$$

   We now want to bound the symmetric distance of the transformed vectors:

   $$d_{Sym}(\texttt{function}(v), \texttt{function}(w)) = \sum_z \left|h_{\texttt{function}(v)}(z) - h_{\texttt{function}(w)}(z)\right|.$$

   Since each function maps each element to boolean $\{\texttt{T}, \texttt{F}\}$, we consider both cases.

   (a) $z = \texttt{T}$:

   $$\left|h_{\texttt{function}(v)}(\texttt{T}) - h_{\texttt{function}(w)}(\texttt{T})\right| = |h_v(\texttt{value}) - h_w(\texttt{value})|$$

(b) $z = \texttt{F}$:

Since any element $z \neq \texttt{value}$ maps to $\texttt{F}$ by definition of $\texttt{is\_equal}$, we consider all elements $z \neq val$: $\left|h_{\texttt{function}(v)}(\texttt{F}) - h_{\texttt{function}(w)}(\texttt{F})\right| = \left|\sum_{z \neq val} h_v(z) - h_w(z)\right|$
By triangle inequality, we have

$$\left|h_{\texttt{function}(v)}(\texttt{F}) - h_{\texttt{function}(w)}(\texttt{F})\right| \leq \sum_{z \neq val} |h_v(z) - h_w(z)|$$

Therefore, we can apply the first two cases respectively to the third line below.

$$\begin{aligned}
d_{Sym}(\texttt{function}(v), \texttt{function}(w)) &= \sum_z \left|h_{\texttt{function}(v)}(z) - h_{\texttt{function}(w)}(z)\right| \\
&= \left|h_{\texttt{function}(v)}(\texttt{T}) - h_{\texttt{function}(w)}(\texttt{T})\right| + \left|h_{\texttt{function}(v)}(\texttt{F}) - h_{\texttt{function}(w)}(\texttt{F})\right| \\
&\leq |h_v(\texttt{value}) - h_w(\texttt{value})| + \sum_{z \neq \texttt{value}} |h_v(z) - h_w(z)| \\
&= \sum_z |h_v(z) - h_w(z)| \\
&= d_{Sym}(v, w)
\end{aligned}$$

Since $d_{Sym}(\texttt{function}(v), \texttt{function}(w)) \leq d_{Sym}(v, w) \leq \texttt{d\_in}$ and $\texttt{d\_in} \leq \texttt{d\_out}$, it follows that the transformations are $\texttt{d\_out}$-close: $d_{Sym}(\texttt{function}(v), \texttt{function}(w)) \leq \texttt{d\_out}$. (grace) TODO in the next round of edits, will use Salil's suggested proof outline with row by row transformation abstraction. This will get rid of casework?

$\square$