fn make_np_sum

Michael Shoemate

May 6, 2025

This proof resides in **"contrib"** because it has not completed the vetting process.

Proves soundness of `make_np_sum` in `__init__.py` at commit f5bb719 (outdated[1]).

`make_np_sum` accepts an `input_domain` and `input_metric`, and returns a stable Transformation that computes the vector-valued sum with bounded L_p sensitivity.

1 Hoare Triple

Preconditions

None

Pseudocode

```

def make_np_sum(input_domain: Domain, input_metric: Metric) -> Transformation:
    """Construct a Transformation that computes a sum over the row axis of a 2-dimensional array.

    :param input_domain: instance of `array2_domain(size=_, num_columns=_)`
    :param input_metric: instance of `symmetric_distance()`

    :return: a Measurement that computes the DP sum
    """
    import opendp.prelude as dp
    np = import_optional_dependency('numpy')

    dp.assert_features("contrib", "floating-point")

    if not str(input_domain).startswith("NPArray2Domain"):  #
        raise ValueError(f"input_domain ({input_domain}) must be NPArray2Domain")  # pragma: no cover

    if input_domain.descriptor.nan:
        raise ValueError(f"input_domain ({input_domain}) must not permit NaN elements")  # pragma: no cover

    if input_metric != dp.symmetric_distance():  #
        raise ValueError("input_metric must be the symmetric distance")

    input_desc = input_domain.descriptor
    norm = input_desc.norm
    if norm is None: #
```

[1]See new changes with `git diff f5bb719..7690fd67 python/src/opendp/extras/numpy/_make_np_sum/__init__.py`

```
27            raise ValueError(f"input_domain ({input_domain}) must have bounds. See make_np_clamp")  # pragma: no cover
28
29        output_metric = {1: dp.l1_distance, 2: dp.l2_distance}[input_desc.p]  #
30
31        if input_desc.size is None:
32            origin = np.atleast_1d(input_desc.origin)
33            norm += np.linalg.norm(origin, ord=input_desc.p)
34            stability = lambda d_in: d_in * norm
35        else:
36            stability = lambda d_in: d_in // 2 * 2 * norm
37
38        return _make_transformation(
39            input_domain,
40            input_metric,
41            dp.vector_domain(dp.atom_domain(T=input_desc.T, nan=False)),
42            output_metric(T=input_desc.T),
43            lambda arg: arg.sum(axis=0),  #
44            stability,
45        )
```

Postcondition

Theorem 1.1. For every setting of the input parameters `(input_domain, input_metric)` to `make_np_sum` such that the given preconditions hold,
`make_np_sum` raises an exception (at compile time or run time) or returns a valid transformation. A valid transformation has the following properties:

1. (Appropriate output domain). For every element x in `input_domain`, `function`(x) is in `output_domain` or raises a data-independent runtime exception.

2. (Stability guarantee). For every pair of elements x, x' in `input_domain` and for every pair (`d_in`, `d_out`), where `d_in` has the associated type for `input_metric` and `d_out` has the associated type for `output_metric`, if x, x' are `d_in`-close under `input_metric`, `stability_map`(`d_in`) does not raise an exception, and `stability_map`(`d_in`) $\leq$ `d_out`, then `function`(x), `function`(x') are `d_out`-close under `output_metric`.

2 Proof

Proof. Let x and x' be datasets that are `d_in`-close with respect to `input_metric`. By 15, the input domain is `NPArray2Domain`, and by 21 input metric is the `SymmetricDistance`. By 26 and 29, the input data has row-p-norm bounded by at most `norm`, which we'll refer to as R, centered at `origin`, which we'll refer to as O.

2.1 Appropriate Output Domain

The input domain consists of 2-dimensional numpy arrays. Since the sum operation eliminates axis zero, only axis one remains, resulting in a 1-dimensional numpy array of the same type. The data loader then reads this numpy array as a vector. Therefore the output is a member of the output vector domain.

2.2 Stability Guarantee

Now consider two cases: when the data set size is known, and when it is not known.

2.2.1 Known Size

Under the floating-point feature flag, we do not consider numerical instability in this analysis. Without numerical instability, addition is commutative and associative, so reordering rows in the data will not affect the outcome. By this logic then, without loss of generality, assume x, x' only differ on the leading $\lfloor d_{in}/2 \rfloor$ rows.

$$\begin{aligned}
&\max_{x \sim x'} d_{Lp}(\text{sum}(x), \text{sum}(x')) && \text{by definition of stability} && (1)\\
&= \max_{x \sim x'} \|\sum_{i=1}^{N} x_i - \sum_{i=1}^{N} x'_i\|_p && \text{substitute the function and metric} && (2)\\
&= \max_{x \sim x'} \|\sum_{i=1}^{\lfloor d_{in}/2 \rfloor} x_i - \sum_{i=1}^{\lfloor d_{in}/2 \rfloor} x'_i\|_p && \text{all trailing terms cancel} && (3)\\
&\leq \max_{x \sim x'} \sum_{i=1}^{\lfloor d_{in}/2 \rfloor} \|x_i - x'_i\|_p && \text{by triangle inequality} && (4)\\
&\leq \sum_{i=1}^{\lfloor d_{in}/2 \rfloor} 2 \cdot R && \text{each row has bounded p-norm of } R && (5)\\
&= \lfloor d_{in}/2 \rfloor \cdot 2 \cdot R && && (6)
\end{aligned}$$

Therefore we've shown that for every pair of elements $x, x' \in$ `input_domain` and every $d_{Sym}(x, x') \leq$ `d_in`, if x, x' are `d_in`-close then `function`(x), `function`(x') are `privacy_map`(`d_in`)-close under `output_metric` (the L^p distance).

2.2.2 Unknown Size

By similar ordering logic as in 2.2.1, then without loss of generality, assume x' has an additional d_{in} trailing rows.

$$\begin{aligned}
&\max_{x \sim x'} d_{Lp}(\text{sum}(x), \text{sum}(x')) && \text{by definition of stability} && (7)\\
&= \max_{x \sim x'} \|\sum_{i=1}^{N} x_i - \sum_{i=1}^{N'} x'_i\|_p && \text{substitute the function and metric} && (8)\\
&= \max_{x \sim x'} \|\sum_{i=1}^{d_{in}} x'_{N+i}\|_p && \text{all but the last terms cancel} && (9)\\
&\leq \max_{x \sim x'} \sum_{i=1}^{d_{in}} \|x'_{N+i}\|_p && \text{by triangle inequality} && (10)\\
&\leq \sum_{i=1}^{d_{in}} (\|O\|_p + R) && \text{each row has bounded p-norm of } R \text{ at } O && (11)\\
&= d_{in} \cdot (\|O\|_p + R) && && (12)
\end{aligned}$$

Therefore we've shown that for every pair of elements $x, x' \in$ `input_domain` and every $d_{Sym}(x, x') \leq$ `d_in`, if x, x' are `d_in`-close then `function`(x), `function`(x') are `privacy_map`(`d_in`)-close under `output_metric` (the L^p distance).

□