fn make_vector_integer_laplace_cks20

Michael Shoemate

February 27, 2024

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

Proves soundness of `make_vector_integer_laplace_cks20` in `mod.rs` at commit f5bb719 (outdated[1]). The function on the resulting measurement takes in a data set `x` (an integer vector), and returns a sample from the Vector Discrete Laplace Distribution centered at `x`, with a fixed noise scale.

PR History

- Pull Request #490

1 Hoare Triple

Preconditions

- Variable `input_domain`, of type `VectorDomain<AtomDomain<T>>`
- Variable `input_metric`, of type `L1Distance<T>`
- Variable `scale`, of type `QO`
- Type `T` must have trait `Integer` and support saturating cast from `IBig` (for postprocessing a noisy big integer back to `T`)
- Type `QO` must have trait `Float` and support casting with controlled rounding from `T` (for converting d_{in} to type `QO`)
- Type `IBig` must be constructable from `T` (to convert the data into a big integer)
- Type `RBig` must be fallibly constructable from `QO` (to convert scale into a rational)

Pseudocode

```python
def make_vector_integer_laplace_cks20(input_domain, input_metric, scale: QO):
    if scale.is_sign_negative():
        raise ValueError("scale must not be negative")

    # conversion to rational will fail if scale is null
    r_scale = RBig.try_from(scale)

    if scale.is_zero():
        def function(x: list[T]):
```

[1] See new changes with `git diff f5bb719..0ede89a5 rust/src/measurements/laplace/discrete/cks20/mod.rs`

```
            return x
    else:
        def function(x: list[T]):
            release = [IBig(x_i) + sample_discrete_laplace(r_scale) for x_i in x]
            # postprocessing
            return [T.saturating_cast(r_i) for r_i in release]

    return Measurement(
        input_domain,
        function,
        input_metric,
        MaxDivergence(QO),
        privacy_map=laplace_map(scale, relaxation=0.)
    )
```

Postcondition

For every setting of the input parameters `(input_domain, input_metric, scale, T, QO)` to `make_vector_integer_laplace_cks20` such that the given preconditions hold, `make_vector_integer_laplace_cks20` raises an exception (at compile time or run time) or returns a valid measurement. A valid measurement has the following property:

1. (Privacy 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_measure`, if x, x' are `d_in`-close under `input_metric`, `privacy_map`(`d_in`) does not raise an exception, and `privacy_map`(`d_in`) $\leq$ `d_out`, then `function`(x), `function`(x') are `d_out`-close under `output_measure`.

2 Proof

Proof. **(Privacy guarantee.)**

The proof assumes the following lemma.

Lemma 2.1. `sample_integer_laplace` and `laplace_map` each satisfy their postcondition.

`sample_integer_laplace` can only fail due to lack of system entropy. This is usually related to the computer's physical environment and not the dataset. The rest of this proof is conditioned on the assumption that this function does not raise an exception.

Let x and x' be datasets that are `d_in`-close with respect to `input_metric`. Here, the metric is `AbsoluteDistance<T>`.

By the postcondition of `sample_integer_laplace`, the output of each call of the function follows the Discrete Laplace Distribution with scale `scale`.

$$
\begin{aligned}
&\max_{x \sim x'} D_\infty(M(x), M(x')) \\
=&\max_{x \sim x'} \max_{z \in supp(M(\cdot))} \ln\left(\frac{\Pr[M(x) = z]}{\Pr[M(x') = z]}\right) && \text{substitute } D_\infty \\
=&\max_{x \sim x'} \max_{z \in \mathbb{Z}} \ln\left(\frac{\Pr[\mathrm{DLap}(x, b) = z]}{\Pr[\mathrm{DLap}(x', b) = z]}\right) && \text{where } b \text{ is the noise scale} \\
=&\max_{x \sim x'} \max_{z \in \mathbb{Z}} \ln\left(\frac{\prod_i^d \frac{\exp^{1/b} - 1}{\exp^{1/b} + 1} \exp\left(-\frac{|x_i - z_i|}{b}\right)}{\prod_i^d \frac{\exp^{1/b} - 1}{\exp^{1/b} + 1} \exp\left(-\frac{|x'_i - z_i|}{b}\right)}\right) && \text{use pdf of Discrete Laplace} \\
=&\max_{x \sim x'} \max_{z \in \mathbb{Z}} \sum_i^d \ln\left(\frac{\exp\left(-\frac{|x_i - z_i|}{b}\right)}{\exp\left(-\frac{|x'_i - z_i|}{b}\right)}\right) && \text{pull product out by log rules} \\
=&\max_{x \sim x'} \max_{z \in \mathbb{Z}} \sum_i^d \frac{|x'_i - z_i| - |x_i - z_i|}{b} && \text{exp and ln cancel} \\
\leq&\frac{\max_{x \sim x'} \sum_i^d |x_i - x'_i|}{b} && \text{by reverse triangle inequality} \\
=&\frac{d_{in}}{b} && \text{by definition of } L^1 \text{ distance}
\end{aligned}
$$

This bound satisfies the postcondition of `laplace_map`. The saturating conversion to `T` is a post-processing step.

Therefore it has been shown that for every pair of elements $x, x' \in$ `input_domain` and every $d_{L1}(x, x') \leq$ `d_in` with `d_in` ≥ 0, if x, x' are `d_in`-close then `function`(x), `function`(x') are `privacy_map(d_in)`-close under `output_measure` (the Max-Divergence). $\square$