`NoisePrivacyMap<L2Distance<RBig>, ZeroConcentratedDivergence> for ZExpFamily<2>`

Michael Shoemate

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

Proves soundness of the implementation of `NoisePrivacyMap` for `ZExpFamily<2>` in `mod.rs` at commit f5bb719 (outdated[1]).

1 Hoare Triple

Precondition

Compiler-Verified

`NoisePrivacyMap` is parameterized as follows:

- `MI`, the input metric, is of type `L2Distance<RBig>`
- `MO`, the output measure, is of type `ZeroConcentratedDivergence`

User-Verified

None

Pseudocode

```
# analogous to impl NoisePrivacyMap<L2Distance<RBig>, ZeroConcentratedDivergence> for
    ZExpFamily<1> in Rust
class ZExpFamily2:
    def noise_privacy_map(
        self, _input_metric: L2Distance[RBig], _output_measure: ZeroConcentratedDivergence
    ) -> PrivacyMap[L2Distance[RBig], ZeroConcentratedDivergence]:
        scale = self.scale
        if scale < RBig.ZERO:  #
            raise "scale must be non-negative"

        def privacy_map(d_in: RBig):
            if d_in < RBig.ZERO:  #
                raise "sensitivity must be non-negative"

            if d_in.is_zero():  #
                return 0.0

            if scale.is_zero():  #
                return float("inf")

```

[1] See new changes with `git diff f5bb719..47bb0dc rust/src/measurements/noise/distribution/gaussian/mod.rs`

```
20            return f64.inf_cast((d_in / scale).pow(2) / rbig(2))  #
21
22        return PrivacyMap.new_fallible(privacy_map)
```

Postcondition

Theorem 1.1. Given a distribution `self`, returns `Err(e)` if `self` is not a valid distribution. Otherwise the output is `Ok(privacy_map)` where `privacy_map` observes the following:

Define `function(x) = x + Z` where `Z` is a vector of iid samples from `self`.

For every pair of elements x, x' in `VectorDomain<AtomDomain<IBig>>`, 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`.

Proof. Line 7 rejects `self` if `self` does not represent a valid distribution, satisfying the error conditions of the postcondition.

We now construct the privacy map. First consider the extreme values of the scale and sensitivity parameters. The sensitivity `d_in`, a bound on distances, must not be negative, as checked on line 11. In the case where sensitivity is zero (line 14), the privacy loss is zero, regardless the choice of scale parameter (even zero). This is because the privacy loss when adjacent datasets are always identical is zero. Otherwise, in the case where the scale is zero, the privacy loss is infinite. To avoid a rational division overflow, line 17 returns infinity.

By line 20, both the sensitivity and scale are positive rationals. Recall Theorem 14 from [CKS20].

Theorem 1.2 (Multivariate Discrete Gaussian Satisfies Concentrated Differential Privacy)**.** Let $\sigma_1, \ldots, \sigma_d > 0$ and $\varepsilon > 0$. Let $q : \mathcal{X}^n \to \mathbb{Z}^d$ satisfy $\sum_{j\in[d]}(q_j(x) - q_j(x'))^2/\sigma_j^2 \leq \varepsilon^2$ for all $x, x' \in \mathcal{X}^n$ differing on a single entry. Define a randomized algorithm $M : \mathcal{X}^n \to \mathbb{Z}^d$ by $M(x) = q(x) + Y$ where $Y_j \leftarrow \mathcal{N}_{\mathbb{Z}}(0, \sigma_j^2)$ independently for all $j \in [d]$. Then M satisfies $\frac{1}{2}\varepsilon^2$-concentrated differential privacy.

Assuming that $\sigma_1, \ldots, \sigma_d = \sigma$, and re-defining $q(x)$ as x and $q(x')$ as x', then if $||x - x'||_2/\sigma \leq \varepsilon$ then M satisfies $\frac{1}{2}\varepsilon^2$-concentrated differential privacy.

Therefore, for all $\alpha > 1$, $D_\alpha(M(x), M(x')) \leq \alpha \cdot (\texttt{d\_in}/\sigma)^2/2$. □

References

[CKS20] Clément L. Canonne, Gautam Kamath, and Thomas Steinke. The discrete gaussian for differential privacy. *CoRR*, abs/2004.00010, 2020.