fn make_canonical_noise

Aishwarya Ramasethu, Yu-Ju Ku, Jordan Awan, Michael Shoemate

April 25, 2025

Proves soundness of make_canonical_noise.

The constructor privatizes a float scalar with noise calibrated to satisfy a fixed privacy guarantee d_out at a fixed sensitivity d_in.

1 Hoare Triple

Preconditions

Compiler-verified

- Argument input_domain of type AtomDomain<f64>.
- Argument input_metric of type AbsoluteDistance<f64>.
- Argument d_in of type f64
- Argument d_out of type (f64, f64) corresponding to epsilon and delta.

Human-verified

None

Pseudocode

```python
def make_canonical_noise(
    input_domain: AtomDomain[f64],
    input_metric: AbsoluteDistance[f64],
    d_in: f64,
    d_out: tuple[f64, f64],
):
    assert not input_domain.nan(), "input data must be non-null" #
    assert not d_in.is_sign_negative() and d_in.is_finite() #

    tradeoff, fixed_point = approximate_to_tradeoff(d_out)
    r_d_in = RBig.try_from(d_in)

    def function(arg: f64) -> f64:  #
        try:  #
            arg = RBig.try_from(arg)
        except Exception:
            arg = RBig(0)

        canonical_rv = CanonicalRV(  #
            shift=arg, scale=r_d_in, tradeoff=tradeoff, fixed_point=fixed_point
        )
        return PartialSample.new(canonical_rv).value()  #
```

```

    def privacy_map(d_in_p: f64) -> f64:  #
        assert 0 <= d_in_p <= d_in
        if d_in == 0:
            return (0.0, 0.0)
        return d_out

    return Measurement.new(
        input_domain,
        function,
        input_metric,
        output_measure=approximate(max_divergence()),
        privacy_map=privacy_map,
    )
```

Postcondition

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

1. (Data-independent runtime errors). For every pair of elements x, x' in `input_domain`, `function`(x) returns an error if and only if `function`(x') returns an error.

2. (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`.

We now prove each part in the postcondition.

Data-independent runtime errors

Proof of Theorem 1.1, Part 1. `PartialSample.value`, hereafter referred to as just `value` (a function) can only fail when the pseudorandom byte generator used in its implementation fails due to lack of system entropy. This is usually related to the computer's physical environment and not the dataset. This is the only source of errors in the function. □

Privacy guarantee

Proving the privacy guarantee of Theorem 1.1 is more involved, and we need to establish several definitions and lemmas first.

In the pseudocode, `d_in` and `d_out` are used to create a tradeoff function. The following defines the corresponding noise distribution.

Definition 1.2 (Awan and Vadhan 2023, Definition 3.7)**.** Let f be a symmetric nontrivial tradeoff function, and let $c \in [0, 1/2)$ be the unique fixed point of f: $f(c) = c$. We define $F_f : \mathbb{R} \to \mathbb{R}$ as

$$F_f(x) = \begin{cases} f(1 - F_f(x+1)) & x < -1/2 \\ c \cdot (1/2 - x) + (1 - c)(x + 1/2) & -1/2 \leq x \leq 1/2 \\ 1 - f(F_f(x-1)) & x > 1/2. \end{cases}$$

For more context on the definition of a tradeoff function, refer to Awan and Vadhan 2023.

We will first prove that the mechanism `function` adds noise from this distribution.

Lemma 1.3. Condition on the assumption that `value` does not raise an exception. Should `value` raise an exception, the function will return an error, as discussed in Proof Part 1.

`function` on Line 13 returns $\texttt{arg} + \texttt{d\_in} \cdot N$ rounded to the nearest `f64` in postprocessing, where N is a sample from some random variable $F_f(\cdot)$, as defined in Definition 1.2, and f is the tradeoff function `tradeoff`.

Proof. The code block on Line 14 converts float `arg` to a rational bignum. Due to 7, the input domain excludes null values, so if the input is a member of the input domain, then the cast will never fail.

Line 8 ensures that `d_in` is well-formed (distances cannot be negative).

`approximate_to_tradeoff` has no user preconditions, so by its postcondition, `tradeoff` is a symmetric nontrivial tradeoff function and `fixed_point` is the fixed-point of `tradeoff`.

On Line 19, by the definition of `CanonicalRV`, `canonical_rv` is a random variable representing $F_f(\cdot)$ (as is defined in Definition 1.2) scaled by `d_in` and shifted by `arg`. That is, `canonical_rv` represents the distribution of $\texttt{arg} + \texttt{d\_in} \cdot N$.

Line 22 then constructs a sampler for the random variable (`PartialSample`) and `.value` draws a sample, rounded to the nearest floating point number. By the postcondition of `PartialSample.value`, the returned value is a post-processing rounding to the nearest float of an infinite-precision sample from the `canonical_rv` random variable. □

Now that we have shown that `function` adds noise from the distribution $F_f(\cdot)$, we can prove that the privacy guarantee is satisfied when noise from $F_f(\cdot)$ is added.

Recall several definitions from Awan and Vadhan 2023. First, we need to define a canonical noise distribution.

Definition 1.4 (Awan and Vadhan 2023, Definition 3.1)**.** Let f be a symmetric nontrivial tradeoff function. A continuous distribution function F is a *canonical noise distribution* (CND) for f if

(1) for every statistic $S : X^n \to \mathbb{R}$ with sensitivity $\Delta > 0$, and $N \sim F(\cdot)$, the mechanism $S(X) + \Delta N$ satisfies f-DP. Equivalently, for every $m \in [0, 1]$, $T(F(\cdot), F(\cdot - m)) \geq f$,

(2) $f(\alpha) = T(F(\cdot), F(\cdot - 1))(\alpha)$ for all $\alpha \in (0, 1)$,

(3) $T(F(\cdot), F(\cdot - 1))(\alpha) = F(F^{-1}(1 - \alpha) - 1)$ for all $\alpha \in (0, 1)$,

(4) $F(x) = 1 - F(-x)$ for all $x \in \mathbb{R}$; that is, F is the cdf of a random variable which is symmetric about zero.

Then, F_f, the noise added in `function`, is a canonical noise distribution:

Theorem 1.5 (Awan and Vadhan 2023, Theorem 3.9)**.** Let f be a symmetric nontrivial tradeoff function and let F_f be as in 1.2. Then F_f is a canonical noise distribution for f.

Using these definitions, and Lemma 1.3, we can prove the privacy guarantee in Theorem 1.1.

Proof of Theorem 1.1 Part 2. Since `tradeoff` is a symmetric nontrivial tradeoff function, and by the definition of `CanonicalRV` that F_f is as in Definition 1.2, then by Theorem 1.5, F_f is a canonical noise distribution for f.

Therefore by Definition 1.4, for every statistic $S : X^n \to \mathbb{R}$ with sensitivity $\Delta > 0$, and $N \sim F(\cdot)$, the mechanism $S(X) + \Delta N$ satisfies f-DP. By Lemma 1.3, `function` returns $S(X) + \Delta N$, where $S(X)$ is `arg`, Δ is `d_in`. Since `tradeoff` is an equivalent but conservative representation of the privacy parameters `d_out`, the mechanism satisfies `d_out`-DP when input datasets may differ by at most `d_in`.

This guarantee is then reflected in the privacy map on line 24. If `d_in_p` is no greater than `d_in`, then the privacy loss is `d_out`.

Therefore, for every pair of elements x, x' in `input_domain` and for every pair $(\texttt{d\_in}, \texttt{d\_out})$, 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 the absolute distance `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`.

$\square$

References

Awan, Jordan and Salil Vadhan (2023). "Canonical Noise Distributions and Private Hypothesis Tests". In: *The Annals of Statistics* 51.2, pp. 547–572.