fn make_randomized_response

Michael Shoemate

September 16, 2025

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

Proves soundness of `make_randomized_response` in `mod.rs` at commit f5bb719 (outdated[1]), a constructor taking a category set `categories` and probability `prob`. The mechanism returned by `make_randomized_response` takes in a data set `arg` (a single category), and...

- ...if `arg` is in `categories`, the mechanism truthfully returns the same value `arg` with probability `prob`, otherwise it lies by selecting one of the other categories uniformly at random.
- ...if `arg` is not in `categories`, it returns a category chosen uniformly at random.

PR History

- Pull Request #490

1 Hoare Triple

Preconditions

- Variable `categories` must be a set with members of type `T`
- Variable `prob` must be of type `f64`

Pseudocode

```python
def make_randomized_response(categories: set[T], prob: f64):
    input_domain = AtomDomain(bool)
    input_metric = DiscreteMetric()
    output_measure = MaxDivergence()

    categories = list(categories)

    if len(categories) < 2:  #
        raise ValueError("expected at least two categories")

    num_categories = len(categories)

    if not (1 / num_categories <= prob <= 1):  #
        raise ValueError("probability must be within [1/num_categories, 1]")

    # prepare constant:
    if prob == 1.0:
```

[1] See new changes with `git diff f5bb719..6535c79 rust/src/measurements/randomized_response/mod.rs`

```
        c = float("inf")
    else:
        c = p.inf_div((1).neg_inf_sub(prob)) \
            .inf_mul(num_categories.inf_sub(1)) \
            .inf_ln()

    def privacy_map(d_in: u32) -> QO:
        if d_in == 0:
            return 0
        else:
            return c

    def function(truth: bool) -> bool:  #
        index = categories.index(truth)
        sample = usize.sample_uniform_int_below(
            len(num_categories) - (0 if index == -1 else 1))

        if index != -1 and sample >= index:
            sample += 1

        lie = categories[sample]

        be_honest = sample_bernoulli_float(prob, false)
        is_member = index != -1
        return truth if be_honest and is_member else lie

    return Measurement(input_domain, function, input_metric, output_measure, privacy_map)
```

Postcondition

For every setting of the input parameters `(categories, prob, T)` to `make_randomized_response` such that the given preconditions hold, `make_randomized_response` raises an error (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 members x and x' in `input_domain`, `invoke`(x) and `invoke`(x') either both return the same error or neither return an error.

2. (Privacy guarantee). For every pair of members x and 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 error, and `privacy_map`(`d_in`) = `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_uniform_int_below` and `sample_bernoulli_float` satisfy their postconditions.

`sample_uniform_int_below` and `sample_bernoulli_float` 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 these functions do not raise an exception.

Let x and x' be datasets that are `d_in`-close with respect to `input_metric`. Here, the metric is `DiscreteMetric` which enforces that `d_in` ≥ 1 if $x \neq x'$ and `d_in` $= 0$ if $x = x'$. If $x = x'$, then the output distributions on x and x' are identical, and therefore the max-divergence is 0.

Now consider the case where $x \neq x'$. For shorthand, we let p represent `prob`, the probability of returning the input, and t denote the number of categories. Note that all categories must be unique as the input data type is a set. This means duplicate categories cannot influence the output distribution.

t must be at least two, by pseudocode line 8, as any fewer would not be useful. p is restricted to $[1/t, 1.0]$ by pseudocode line 13, as any less would not be useful.

We'll first consider all possible output probabilities, and then use this to upper bound the ratio of any two probabilities. For any outcome $y \in$ `candidates`, the probability of observing y is one of three values:

1. When the mechanism is honest:
$$\Pr[M(x) = y | y = x] = p$$

2. When the mechanism lies:
$$\Pr[M(x) = y | y \neq x \wedge x \in \texttt{candidates}] = \frac{1-p}{t-1}$$

3. When the input is not in the category set, the output is uniformly sampled from the candidates:
$$\Pr[M(x) = y | y \neq x \wedge x \notin \texttt{candidates}] = \frac{1}{t}$$

Lemma 2.2. The probability of case 3 is bounded by cases one and two:
$$\frac{1-p}{t-1} \leq \frac{1}{t} \leq p \tag{1}$$

Proof. $1/t$ is bounded above by case one (p) due to pseudocode line 13. Reusing 13, $\frac{1-p}{t-1} \leq \frac{1-1/t}{t-1} = \frac{1}{t}$. Therefore $1/t$ is also bounded below by case two ($\frac{1-p}{t-1}$). □

By 2.2, the divergence is never maximized when the input is not in the category set, which simplifies the following analysis.

We now consider the max-divergence of the mechanism over all choices of neighboring datasets.

$$\begin{aligned}
&\max_{x \sim x'} D_\infty(M(x), M(x')) && (2)\\
=&\max_{x \sim x'} \max_{S \subseteq Supp(M(x))} \ln\left(\frac{\Pr[M(x) \in S]}{\Pr[M(x') \in S]}\right) && (3)\\
\leq&\max_{x \sim x'} \max_{y \in Supp(M(x))} \ln\left(\frac{\Pr[M(x) = y]}{\Pr[M(x') = y]}\right) && \text{Lemma 3.3 [1]} \quad (4)\\
=&\ln\left(\max\left(\frac{p \cdot (t-1)}{1-p}, \frac{(1-p)\cdot(t-1)}{p}, \frac{(1-p)\cdot(t-1)}{(1-p)\cdot(t-1)}\right)\right) && (5)\\
=&\ln\left(\frac{p \cdot (t-1)}{1-p}\right) && (6)
\end{aligned}$$

The terms in the maximum on line 5 cover all combinations of x, x' and y. Respectively:

1. When $y = x$.
2. When $y \neq x$ and $y = x'$.

3. When $y \neq x$ and $y \neq x'$.

Pseudocode line 16 implements this bound with conservative rounding towards positive infinity. When `d_in` > 0 and no exception is raised in computing `c` $=$ `privacy_map(d_in)`, then $\ln\left(\frac{p\cdot(t-1)}{1-p}\right) \leq$ `c`.

Therefore it has been shown that for every pair of elements $x, x' \in$ `input_domain` and every $d_{DM}(x, x') \leq$ `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). $\square$

References

[1] Shiva P. Kasiviswanathan and Adam Smith. On the "semantics" of differential privacy: A bayesian formulation. *Journal of Privacy and Confidentiality*, 6(1), June 2014.