

fn make_base_discrete_exponential

Michael Shoemate

April 12, 2023

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

Proves soundness of `make_base_discrete_exponential` in `mod.rs` at commit `f5bb719` (outdated¹). `make_base_discrete_exponential` returns a `Measurement` that noisily selects the index of the greatest score, from a vector of input scores.

This released index can be later be used to index into a public candidate set (postprocessing).

Vetting History

- [Pull Request #456](#)

The naive implementation samples some index k from a categorical distribution, with probabilities assigned to each candidate relative to their score. We may use inverse transform sampling to select the smallest index k for which the cumulative probability is greater than some $U \sim \text{Uniform}(0, 1)$.

$$M(s) = \underset{i}{\operatorname{argmin}} \sum_{i=1}^k p_i \geq U \quad (1)$$

The probability of index k being selected is the normalization of its likelihood $e^{s_k/\tau}$. As a candidate's score s_k increases, the candidate becomes exponentially more likely to be selected.

$$p_k = \frac{e^{s_k/\tau}}{\sum_i e^{s_i/\tau}} \quad (2)$$

This equation introduces a new temperature parameter, τ , which calibrates how distinguishable scores are from each other. As temperature increases, the categorical output distribution tends towards entropy/uniformity and becomes more privacy preserving. As temperature decreases, the categorical distribution tends towards a one-hot vector, becoming less private. Temperature is related to ϵ and the sensitivity (Δ) of the scoring function as follows:

$$\tau = \Delta/\epsilon \quad (3)$$

When ϵ increases, temperature decreases, and candidates become more distinguishable from each other. We also divide scores by their global sensitivity to normalize the sensitivity to one. In the differential privacy literature for the exponential mechanism, the sensitivity is often multiplied by two. In OpenDP this factor is bundled into the Δ term, which is expressed in terms of a metric that captures monotonicity.

¹See new changes with `git diff f5bb719..03c4ada5 rust/src/measurements/discrete_exponential/mod.rs`

1 Gumbel Reparameterization

In practice, computing $e^{s_i/\tau}$ is prone to zero underflow and overflow. Specifically, a scaled score of just -709 underflows to zero and $+710$ overflows to infinity when stored in a 64-bit float. A simple improvement is to shift the scores by subtracting the greatest score from all scores. In idealized arithmetic, the resulting probabilities are not affected by shifts in the underlying scores. On finite data types, this shift prevents a catastrophic overflow, but makes underflow more likely, causing tail values of the distribution to round to zero.

The inverse transform sampling is also subject to accumulated rounding errors from the arithmetic and sum, which influence the likelihood of being chosen.

The Gumbel-max trick may instead be used to privately select an index. Let $K = \operatorname{argmax}_k G_k$, a random variable representing the selected index. Denote the k^{th} noisy score as $G_k \sim \text{Gumbel}(\mu = s_k/\tau)$. K can be sampled via an inverse transform, where u_k is sampled iid uniformly from $(0, 1)$:

$$M(s) = \operatorname{argmax}_k (s_k/\tau - \log(-\log(u_k))) \quad (4)$$

Theorem 1.1. Sampling from K is equivalent to sampling from the softmax, because $P(K = k) = p_k$. [1]

$$\begin{aligned}
 P(K = k) &= P(G_k = \max_i G_i) && \text{by definition of } K \\
 &= P(-\log(Z_k/N) = \max_i -\log(Z_i/N)) && \text{by 1.2} \\
 &= P(\log(Z_k/N) = \min_i \log(Z_i/N)) && \text{since } \max - a_i = -\min_i a_i \\
 &= P(Z_k = \min_i Z_i) && \text{simplify monotonic terms} \\
 &= P(Z_k \leq \min_{i \neq k} Z_i) \\
 &= P(Z_k \leq Q) && \text{by 1.3 where } Q \sim \text{Exp}(\sum_{i \neq k} p_i) \\
 &= \frac{p_k}{p_k + \sum_{i \neq k} p_i} && \text{by 1.4} \\
 &= p_k && \text{since } p_k + \sum_{i \neq k} p_i = 1
 \end{aligned}$$

Lemma 1.2. $G_k = -\log(Z_k/N)$ where $Z_k \sim \text{Exp}(p_k)$ and normalization term $N = \sum_i e^{s_i/\tau}$.

$$\begin{aligned}
 G_k &= s_k/\tau - \log(-\log(U_k)) && \text{Gumbel PDF centered at } s_k/\tau \\
 &= \log(e^{s_k/\tau}) - \log(-\log(U_k)) \\
 &= \log(p_k N) - \log(-\log(U_k)) && \text{since } p_k = e^{s_k/\tau}/N \\
 &= \log(p_k N / (-\log(U_k))) \\
 &= -\log(-\log(U_k) / (p_k N)) \\
 &= -\log(Z_k/N) && \text{substitute } Z_k = -\log(U_k)/p_k
 \end{aligned}$$

Lemma 1.3. If $X_1 \sim \text{Exp}(\lambda_1)$, $X_2 \sim \text{Exp}(\lambda_2)$ and $Z \sim \text{Exp}(\lambda_1 + \lambda_2)$, then $\min(X_1, X_2) \sim Z$.

$$\begin{aligned}
 P(\min(X_1, X_2) \geq x) &= P(X_1 \geq x)P(X_2 \geq x) && \text{by independence} \\
 &= e^{-\lambda_1 x} e^{-\lambda_2 x} && \text{substitute exponential density} \\
 &= e^{-(\lambda_1 + \lambda_2)x} \\
 &= P(Z \geq x) && \text{substitute exponential density}
 \end{aligned}$$

Lemma 1.4. If $X_1 \sim \text{Exp}(\lambda_1)$, $X_2 \sim \text{Exp}(\lambda_2)$, then $P(X_1 \leq X_2) = \frac{\lambda_1}{\lambda_1 + \lambda_2}$.

$$\begin{aligned} P(X_1 \leq X_2) &= \int_0^\infty \int_{x_1}^\infty \lambda_1 \lambda_2 e^{-\lambda_1 x_1} e^{-\lambda_2 x_2} dx_1 dx_2 \\ &= \int_0^\infty -\lambda e^{-(\lambda_1 + \lambda_2)x_1} dx_1 \\ &= \frac{\lambda_1}{\lambda_1 + \lambda_2} \end{aligned}$$

1.1 Metric

We need a metric that captures the distance between score vectors u and v respectively on neighboring datasets. The i^{th} element of each score vector is the score for the i^{th} candidate. Traditionally, the sensitivity of the scoring function is measured in terms of some metric **InfDistance**, the greatest that any one score may change (or more formally, the L_∞ norm on distances). The sensitivity is defined as follows:

$$\Delta_\infty = \max_{u \sim v} \max_i |s_i - s'_i| \quad (5)$$

Unfortunately, this choice of metric always results in a loosening by a factor of 2 when evaluating the privacy guarantee of the exponential mechanism. This is because both the i^{th} likelihood and normalization term may vary in opposite directions, resulting in a more distinguishing event. However, this loosening is not necessary if we can prove that the scoring function is monotonic, because the i^{th} likelihood and normalization term will always vary in the same direction.

We instead use a slight adjustment to this metric, **InfDifferenceDistance**, with sensitivity as follows:

$$\Delta_{IDD} = \max_{u \sim v} \max_{ij} |(u_i - v_i) - (u_j - v_j)| \quad (6)$$

Consider when the scoring function is not monotonic. The sensitivity is maximized when $u_i - v_i$ and $u_j - v_j$ vary maximally in opposite directions, resulting in the same loosening factor of 2. On the other hand, when the scoring function is monotonic, the sign of the $u_i - v_i$ term matches the sign of the $u_j - v_j$ term, and their magnitudes cancel. Therefore, when the scorer is monotonic, the sensitivity is maximized when one term is zero. It is shown in 3.2 that a tighter analysis of the exponential mechanism is compatible with a score vector whose sensitivity is expressed in terms of this metric.

2 Hoare Triple

Precondition

- TIA (input atom type) is a type with traits **Number** and **CastInternalRational**
- QO (output distance type) is a type with traits **Float**, **CastInternalRational** and **DistanceConstant** from type TIA

Function

```
1 def make_base_discrete_exponential(temperature: TIA, optimize: Optimize):
2     if temperature <= 0:
3         raise ValueError("temperature must be positive")
4
5     if optimize == "max":
6         sign = +1
7     elif optimize == "min":
```

```

8     sign = -1
9     else:
10         raise ValueError("must specify optimization")
11
12     temp_frac = Fraction(temperature)
13
14     def function(scores: List[TIA]):
15         def map_gumbel(score):
16             return GumbelPSRN(shift=sign * Fraction(score) / temp_frac)
17         gumbel_scores = map(map_gumbel, scores)
18
19         def reduce_best(a, b):
20             return a if a[1].greater_than(b[1]) else b
21         return reduce(reduce_best, enumerate(gumbel_scores))[0]
22
23     def privacy_map(d_in: TIA):
24         d_in = Q0.inf_cast(d_in)
25         if d_in < 0:
26             raise ValueError("input distance must be non-negative")
27
28         if d_in == 0:
29             return 0
30
31         return d_in.inf_div(temperature)
32
33     return Measurement(
34         input_domain=VectorDomain(AtomDomain(TIA)),
35         function=function,
36         input_metric=InfDifferenceDistance(TIA),
37         output_metric=MaxDivergence(Q0),
38         privacy_map=privacy_map,
39     )

```

Postcondition

For every setting of the input parameters `temperature` to `make_base_discrete_exponential` such that the given preconditions hold, `make_base_discrete_exponential` raises an exception (at compile time or run time) or returns a valid measurement. A valid measurement has the following properties:

1. (Domain-metric compatibility). The domain `input_domain` matches one of the possible domains listed in the definition of `input_metric`.
2. (Privacy guarantee). For every pair of elements u, v 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 u, v are d_{in} -close under `input_metric` and `privacy_map`(d_{in}) $\leq d_{out}$, then `function`(u), `function`(v) are d_{out} -close under `output_measure`.

3 Proof

3.1 Domain-metric compatibility

`InfDifferenceDistance` is well-defined on `VectorDomain`<`AtomDomain`<`TIA`>> when `TIA` implements the trait bounds specified in the preconditions.

3.2 Privacy Guarantee

Lemma 3.1. Assume u, v in `input_domain`. Then $\ln \left(\frac{\sum_i \exp(\frac{\epsilon v_i}{\Delta})}{\sum_i \exp(\frac{\epsilon u_i}{\Delta})} \right) \leq \frac{\epsilon \max_j (v_j - u_j)}{\Delta}$.

Proof.

$$\begin{aligned}
\ln \left(\frac{\sum_i \exp\left(\frac{\epsilon v_i}{\Delta}\right)}{\sum_i \exp\left(\frac{\epsilon u_i}{\Delta}\right)} \right) &= \ln \left(\frac{\sum_i \exp\left(\frac{\epsilon(v_i - u_i + u_i)}{\Delta}\right)}{\sum_i \exp\left(\frac{\epsilon u_i}{\Delta}\right)} \right) \\
&= \ln \left(\frac{\sum_i \exp\left(\frac{\epsilon(v_i - u_i)}{\Delta}\right) \exp\left(\frac{\epsilon u_i}{\Delta}\right)}{\sum_i \exp\left(\frac{\epsilon u_i}{\Delta}\right)} \right) \\
&\leq \ln \left(\frac{\exp\left(\frac{\epsilon \max_j (v_j - u_j)}{\Delta}\right) \sum_i \exp\left(\frac{\epsilon u_i}{\Delta}\right)}{\sum_i \exp\left(\frac{\epsilon u_i}{\Delta}\right)} \right) \\
&= \frac{\epsilon \max_j (v_j - u_j)}{\Delta}
\end{aligned}$$

□

Assume u, v in `input_domain` are `d_in`-close under `InfDifferenceDistance` and `privacy_map(d_in) ≤ d_out`.

$$\begin{aligned}
&\max_{u \sim v} D_\infty(\text{function}(u), \text{function}(v)) \\
&= \max_{u \sim v} \max_i \ln \left(\frac{\Pr[\text{function}(u) = i]}{\Pr[\text{function}(v) = i]} \right) && \text{by MaxDivergence} \\
&= \max_{u \sim v} \max_i \ln \left(\frac{\Pr[\text{argmax}_k (u_k/\tau - \ln(-\ln(U_k))) = i]}{\Pr[\text{argmax}_k (v_k/\tau - \ln(-\ln(U_k))) = i]} \right) && \text{substitute function}
\end{aligned}$$

Assuming `privacy_map(d_in) ≤ d_out = ε`, then $\tau \geq \Delta/\epsilon$.

$$\begin{aligned}
&\leq \max_{u \sim v} \max_i \ln \left(\frac{\Pr[\text{argmax}_k (u_k \epsilon/\Delta - \ln(-\ln(U_k))) = i]}{\Pr[\text{argmax}_k (v_k \epsilon/\Delta - \ln(-\ln(U_k))) = i]} \right) \\
&= \max_{u \sim v} \max_i \ln \left(\frac{\exp\left(\frac{\epsilon u_i}{\Delta}\right)}{\sum_k \exp\left(\frac{\epsilon u_k}{\Delta}\right)} \bigg/ \frac{\exp\left(\frac{\epsilon v_i}{\Delta}\right)}{\sum_k \exp\left(\frac{\epsilon v_k}{\Delta}\right)} \right) && \text{by 1} \\
&= \max_{u \sim v} \max_i \ln \left(\frac{\exp\left(\frac{\epsilon u_i}{\Delta}\right) \sum_k \exp\left(\frac{\epsilon v_k}{\Delta}\right)}{\exp\left(\frac{\epsilon v_i}{\Delta}\right) \sum_k \exp\left(\frac{\epsilon u_k}{\Delta}\right)} \right) \\
&= \max_{u \sim v} \max_i \ln \left(\frac{\exp\left(\frac{\epsilon u_i}{\Delta}\right)}{\exp\left(\frac{\epsilon v_i}{\Delta}\right)} \right) + \ln \left(\frac{\sum_k \exp\left(\frac{\epsilon v_k}{\Delta}\right)}{\sum_k \exp\left(\frac{\epsilon u_k}{\Delta}\right)} \right) \\
&= \max_{u \sim v} \max_i \frac{\epsilon(u_i - v_i)}{\Delta} + \ln \left(\frac{\sum_k \exp\left(\frac{\epsilon v_k}{\Delta}\right)}{\sum_k \exp\left(\frac{\epsilon u_k}{\Delta}\right)} \right) \\
&\leq \max_{u \sim v} \max_i \frac{\epsilon(u_i - v_i)}{\Delta} + \frac{\epsilon \max_j (v_j - u_j)}{\Delta} && \text{by 3.1} \\
&\leq \frac{\max_{ij} |(u_i - v_i) - (u_j - v_j)|}{\Delta} \\
&\leq \epsilon && \text{by InfDifferenceDistance} \\
&= \text{d_out}
\end{aligned}$$

It has been shown that `function(u)` and `function(v)` are `d_out`-close under `output_measure` under the definitions of `function` and `privacy_map`, and the conditions on the input distance and privacy map.

References

- [1] Andrés Muñoz Medina and Jennifer Gillenwater. Duff: A dataset-distance-based utility function family for the exponential mechanism. *ArXiv*, abs/2010.04235, 2020.