fn make_base_exponential_candidates_gumbel

Michael Shoemate

March 10, 2023

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

This implementation is susceptible to floating-point vulnerabilities.

Proves soundness of `make_base_exponential_candidates_gumbel` in `mod.rs` at commit f5bb719 (outdated[1]). `make_base_exponential_candidates_gumbel` 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 $U \sim Uniform(0, 1)$.

$$M(s) = argmin_k \sum_i^k p_i >= U \tag{1}$$

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.

$$p_k = \frac{e^{s_k/\tau}}{\sum_i e^{s_i/\tau}} \tag{2}$$

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:

$$\tau = \Delta/\epsilon \tag{3}$$

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.

[1] See new changes with `git diff f5bb719..931ff520 rust/src/measurements/exponential_candidates_gumbel/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 $K = argmax_k G_k$, a random variable representing the selected index. Denote the k^{th} noisy score as $G_k \sim Gumbel(\mu = s_k/\tau)$. K can be sampled via an inverse transform, where u_k is sampled iid uniformly from $(0, 1)$:

$$M(s) = argmax_k(s_k/\tau - log(-log(u_k))) \tag{4}$$

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

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

Lemma 1.2. $G_k = -log(Z_k/N)$ where $Z_k \sim Exp(p_k)$ 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 $X_1 \sim Exp(\lambda_1)$, $X_2 \sim Exp(\lambda_2)$ and $Z \sim Exp(\lambda_1 + \lambda_2)$, 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 $X_1 \sim Exp(\lambda_1)$, $X_2 \sim Exp(\lambda_2)$, 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 s and s' 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 `InfDistance`, the greatest that any one score may change (or more formally, the L_∞ norm on distances). The sensitivity is defined as follows:

$$\Delta_\infty = \max_{s \sim s'} \max_i |s_i - s'_i| \tag{5}$$

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:

$$\Delta_{\infty'} = \max_{s \sim s'} \max_{ij} \left|(s_i - s'_i) - (s_j - s'_j)\right| \tag{6}$$

Consider when the scoring function is not monotonic. The sensitivity is maximized when $s_i - s'_i$ and $s_j - s'_j$ 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 $s_i - s'_i$ term matches the sign of the $s_j - s'_j$ 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 trait `Number`.
- `QO` (output distance type) is a type with traits `Float`, `InfCast` from type `TIA`, `RoundCast` from type `TIA`, and `SampleUniform`

Function

```
def make_base_exponential_candidates_gumbel(temperature: TIA):
    if temperature <= 0 {
        raise ValueError("temperature must be positive")
    }
    def function(scores: List[TIA]):
        noisy_score = lambda i: scores[i] / temperature -ln( -ln(sample_uniform()))
        return max(range(len(scores)), key=noisy_score)
```

```
    def privacy_map(d_in: TIA):
        d_in = QO.inf_cast(d_in)
        if d_in < 0 {
            raise ValueError("input distance must be non-negative")
        }
        if d_in == 0 {
            return 0
        }
        if temperature == 0 {
            return QO("inf")
        }
        return d_in.inf_div(temperature)

    return Measurement(
        input_domain=VectorDomain(AllDomain(TIA)),
        function=function,
        input_metric=InfDifferenceDistance(TIA),
        output_metric=MaxDivergence(QO),
        privacy_map=privacy_map,
    )
```

Postcondition

For every setting of the input parameters `temperature` to `make_base_exponential_candidates_gumbel` such that the given preconditions hold, `make_base_exponential_candidates_gumbel` 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<AllDomain<TIA»` when `TIA` implements the trait bounds specified in the preconditions.

3.2 Privacy Guarantee

A mechanism M satisfies ϵ-differential privacy if, for any adjacent score vectors s and s' and for all $K \subseteq Range(M)$

$$P(M(s) \in K) \leq e^{\epsilon} P(M(s') \in K) \tag{7}$$

which implies

$$\ln\left(\frac{P(M(s) \in K)}{P(M(s') \in K)}\right) \leq \epsilon \tag{8}$$

The following proves that M as defined in 1 satisfies ϵ-differential privacy.

$$
\begin{aligned}
\ln\left(\frac{P(M(s)=k)}{P(M(s')=k)}\right) &= \ln\left(\frac{\exp\left(\frac{\epsilon s_k}{\Delta}\right)}{\sum_i \exp\left(\frac{\epsilon s_i}{\Delta}\right)} \Big/ \frac{\exp\left(\frac{\epsilon s'_k}{\Delta}\right)}{\sum_i \exp\left(\frac{\epsilon s'_i}{\Delta}\right)}\right) && \text{substitute 2, assuming } \tau \geq \Delta/\epsilon \\
&= \ln\left(\frac{\exp\left(\frac{\epsilon s_k}{\Delta}\right)}{\exp\left(\frac{\epsilon s'_k}{\Delta}\right)} \frac{\sum_i \exp\left(\frac{\epsilon s'_i}{\Delta}\right)}{\sum_i \exp\left(\frac{\epsilon s_i}{\Delta}\right)}\right) \\
&= \ln\left(\frac{\exp\left(\frac{\epsilon s_k}{\Delta}\right)}{\exp\left(\frac{\epsilon s'_k}{\Delta}\right)}\right) + ln\left(\frac{\sum_i \exp\left(\frac{\epsilon s'_i}{\Delta}\right)}{\sum_i \exp\left(\frac{\epsilon s_i}{\Delta}\right)}\right) \\
&= \frac{\epsilon(s_k - s'_k)}{\Delta} + ln\left(\frac{\sum_i \exp\left(\frac{\epsilon s'_i}{\Delta}\right)}{\sum_i \exp\left(\frac{\epsilon s_i}{\Delta}\right)}\right) \\
&\leq \frac{\epsilon(s_k - s'_k)}{\Delta} + \frac{-\epsilon \max_j (s_j - s'_j)}{\Delta} && \text{by 3.1} \\
&\leq \frac{\epsilon \max_{ij} \left|(s_i - s'_i) - (s_j - s'_j)\right|}{\Delta} \\
&\leq \epsilon
\end{aligned}
$$

Lemma 3.1. $ln\left(\frac{\sum_i \exp\left(\frac{\epsilon s'_i}{\Delta}\right)}{\sum_i \exp\left(\frac{\epsilon s_i}{\Delta}\right)}\right) \geq \frac{\epsilon \max_j (s_j - s'_j)}{\Delta}$.

$$
\begin{aligned}
ln\left(\frac{\sum_i \exp\left(\frac{\epsilon s'_i}{\Delta}\right)}{\sum_i \exp\left(\frac{\epsilon s_i}{\Delta}\right)}\right) &= ln\left(\frac{\sum_i \exp\left(\frac{\epsilon(s'_i - s_i + s_i)}{\Delta}\right)}{\sum_i \exp\left(\frac{\epsilon s_i}{\Delta}\right)}\right) \\
&= ln\left(\frac{\sum_i \exp\left(\frac{\epsilon(s'_i - s_i)}{\Delta}\right) \exp\left(\frac{\epsilon(s_i)}{\Delta}\right)}{\sum_i \exp\left(\frac{\epsilon s_i}{\Delta}\right)}\right) \\
&\geq ln\left(\frac{\exp\left(\frac{\epsilon min_j(s'_j - s_j)}{\Delta}\right) \sum_i \exp\left(\frac{\epsilon(s_i)}{\Delta}\right)}{\sum_i \exp\left(\frac{\epsilon s_i}{\Delta}\right)}\right) \\
&= \frac{\epsilon min_j(s'_j - s_j)}{\Delta} \\
&= -\frac{\epsilon \max_j (s_j - s'_j)}{\Delta}
\end{aligned}
$$

By 1, then the Gumbel sampling approach as implemented in the function would also satisfy ϵ-differential privacy, if it were implemented with infinite-precision arithmetic.

Since the implementation makes use of finite floating-point numbers, this implementation is kept behind the "floating-point" compilation flag.

A floating-point safe implementation is pending in Pull Request #653.

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.