`fn make_permute_and_flip`

Tudor Cebere

This document proves soundness of `make_permute_and_flip` [3] in `mod.rs` at commit e62b0aa2 (outdated[1]). `make_permute_and_flip` returns a Measurement that noisily selects the index of the greatest score from a vector of input scores. This released index can later be used to index into a public candidate set (postprocessing).

It is appealing to implement permute and flip due to its discrete nature. The algorithm requires two primitives that need to be carefully implemented to work out using floating point arithmetics:

1. A shuffling algorithm to shuffle the order in which the elements are being analysed. This is implemented via the Fisher-Yates shuffling algorithm, and it requires uniformly selecting an integer from a set, implemented using SampleIntBelow.

2. A exact bernoulli sampler for distributions of the form $Bern(exp(-\gamma))$. The naive technique would require computing the binary expansions of $exp(-\gamma)$. We bypass this issue by using the exact sampler presented in section 5.2 in [1], essentially reducing sampling $Bern(exp(-\gamma))$ to sampling $Bern(\frac{\gamma}{k})$, $k \in N_+$, for which Bernoulli factories are known.

Permute and flip is equivalent to report noisy max with exponential noise [2]. Implementation-wise, we will follow permute-and-flip, yet proving the correctness of the algorithm will be done via this equivalence.

Metric

Permute and flip reuses the same metric as `RNM-gumbel`.

1 Hoare Triple

Preconditions

- `TIA` (input atom type) is a type with traits `Number` and `CastInternalRational`

Pseudocode

```
def function(scores: list[TIA]):

    best_score = max(scores)
    indexes = range(len(scores))
    shuffled_indexes = shuffle(indexes)

    for current_index in shuffled_indexes:
        coin_bias = (best_score - scores[current_index])/scale
        if Bern(coin_bias):
            return current_index
```

[1]See new changes with `git diff e62b0aa2..841450a7 rust/src/measurements/noisy_top_k/exponential/mod.rs`

Postcondition

For every setting of the input parameters `input_domain, input_metric, scale, optimize, TIA` to `permute_and_flip` such that the given preconditions hold, `permute_and_flip` 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

Before proving the privacy guarantees, we state a few required definitions and lemmas:

Definition 2.1. Report noisy max with exponential noise computes the index of the maximum element from a set of candidates $u \in$ `d_in`, add isotropic exponential noise $Z_i \sim Exp(1/\lambda)$ to each element in the candidate set u and returns the maximum index as follows:

$$\texttt{RNM-Exp}(u) = \text{argmax}_i(u_i + Z_i), Z_i \sim \text{Exp}(1/\lambda) \tag{1}$$

Lemma 2.2. The permute-and-flip mechanism is equivalent to the report-noisy-max with exponential noise mechanism. Providing proof for this is out of the scope of this document; the details on this equivalence are presented in [2]. The pseudocode for report noisy max with exponential noise is implemented in Algorithm **??** for completeness.

Lemma 2.3. Let $X_1, X_2 \sim Exp(\lambda), \Delta \geq 0$, then

$$Pr[X_1 - X_2 \geq \Delta] = e^{-\Delta\lambda} Pr[X_1 - X_2 \geq 0] \tag{2}$$

Proof.

$$Pr[X_1 - X_2 \geq \Delta] \tag{3}$$
$$= 1 - Pr[X_1 \leq \Delta + X_2] \tag{4}$$
$$= 1 - \int_0^\infty Pr[X_1 \leq \Delta + X_2 | X_2 = x] Pr[X_2 = x] dx \qquad \text{by Law of Total Probability} \tag{5}$$
$$= 1 - \int_0^\infty Pr[X_1 \leq \Delta + x] Pr[X_2 = x] dx \qquad \text{by the fact that } \Delta > 0 \tag{6}$$
$$= 1 - \int_0^\infty \lambda(1 - e^{-(x+\Delta)\lambda}) e^{-x\lambda} dx \tag{7}$$
$$= 1 - \lambda \int_0^\infty e^{-x\lambda} dx + \lambda e^{-\Delta\lambda} \int_0^\infty e^{-2x\lambda} dx \tag{8}$$
$$= 1 - 1 + e^{-\Delta\lambda}/2 \qquad Pr[X_1 - X_2 \leq 0] = Pr[X_1 - X_2 \geq 0] = 1/2 \tag{9}$$
$$= e^{-\Delta\lambda} Pr[X_1 - X_2 \geq 0] \tag{10}$$

□

Lemma 2.4. Let $u, v \in$ `input_domain` be two vector scores, where each score is associated with input neighbouring datasets D, respectively D'. Assume u, v in `input_domain` are `d_in`-close under `LInfDistance` and `privacy_map`(`d_in`) $\leq$ `d_out`. Let $Z^* = min_{Z_i}\{u_i + Z_i \geq u_j + Z_j\}, \forall i \neq j$. Then

$$\ln\left(\frac{Pr[\texttt{function}(u) = i]}{Pr[\texttt{function}(v) = i]}\right) = \ln\left(\frac{Pr[Z_i \geq Z^*]}{Pr[Z_i \geq Z^* + \texttt{d\_in}]}\right) \tag{11}$$

Proof.

$$\begin{aligned}
&\ln\left(\frac{Pr[\texttt{function}(u) = i]}{Pr[\texttt{function}(v) = i]}\right) && && (12)\\
&= \ln\left(\frac{Pr[\texttt{RNM-Exp(u)} = i]}{Pr[\texttt{RNM-Exp}(v) = i]}\right) && \text{by Lemma 2.2} && (13)\\
&= \ln\left(\frac{Pr[\text{argmax}_k(u_k + Z_k) = i]}{Pr[\text{argmax}_k(v_k + Z_k) = i]}\right) && \text{by Definition 2.1} && (14)
\end{aligned}$$

Observe that for a fixed i, report noisy max outputs i if:

$$\begin{aligned}
&u_i + Z^* \geq u_j + Z_j, \forall i \neq j && \iff && (15)\\
&u_i + (v_i - v_i) + Z^* \geq u_j + (v_j - v_j) + Z_j && \iff && (16)\\
&v_i + (u_i - v_i) + Z^* \geq v_j + (u_j - v_j) + Z_j && \iff && (17)\\
&v_i + ((u_i - v_i) - (u_j - v_j) + Z^*) \geq v_j + Z_j && \iff && (18)\\
&v_i + (\Delta + Z^*) \geq v_j + Z_j && && (19)
\end{aligned}$$

In other words, if $Z_i \geq (\Delta + Z^*)$, then `function`(u) = `function`$(v) = i$. This yields us:

$$\ln\left(\frac{Pr[\text{argmax}_k(u_k + Z_k) = i]}{Pr[\text{argmax}_k(v_k + Z_k) = i]}\right) = \ln\left(\frac{Pr[Z_i \geq Z^*]}{Pr[Z_i \geq \Delta + Z^*]}\right) \tag{20}$$

□

Theorem 2.5. The permute-and-flip mechanism (a.k.a `function`) satisfies $(\epsilon, 0)$-DP

Proof.

$$\begin{aligned}
&\max_{u\sim v} D_\infty(M(u)|M(v)) && && (21)\\
&= \max_{u\sim v} max_i \ln\left(\frac{\Pr[function(u) = i]}{\Pr[function(v) = i]}\right) && && (22)\\
&= \max_{u\sim v} max_i \ln\left(\frac{Pr[\texttt{RNM-Exp(u)} = i]}{Pr[\texttt{RNM-Exp}(v) = i]}\right) && \text{by Lemma 2.2} && (23)\\
&= \max_{u\sim v} max_i \ln\left(\frac{Pr[\text{argmax}_k(u_k + Z_k) = i]}{Pr[\text{argmax}_k(v_k + Z_k) = i]}\right) && \text{by Definition 2.1} && (24)\\
&= \max_{u\sim v} max_i \ln\left(\frac{Pr[Z_i \geq Z^*]}{Pr[Z_i \geq Z^* + \texttt{d\_in}]}\right) && \text{by Lemma 2.4} && (25)\\
&\leq \frac{\texttt{d\_in}}{\lambda} && \text{by Lemma 2.3} && (26)\\
&= \texttt{d\_out} && \text{by setting } \lambda = \frac{\texttt{d\_in}}{\texttt{d\_out}} && (27)
\end{aligned}$$

□

References

[1] Clément L Canonne, Gautam Kamath, and Thomas Steinke. The discrete gaussian for differential privacy. *Advances in Neural Information Processing Systems*, 33:15676–15688, 2020.

[2] Zeyu Ding, Daniel Kifer, Thomas Steinke, Yuxin Wang, Yingtai Xiao, Danfeng Zhang, et al. The permute-and-flip mechanism is identical to report-noisy-max with exponential noise. *arXiv preprint arXiv:2105.07260*, 2021.

[3] Ryan McKenna and Daniel R Sheldon. Permute-and-flip: A new mechanism for differentially private selection. *Advances in Neural Information Processing Systems*, 33:193–203, 2020.