fn make_quantile_score_candidates

Michael Shoemate Christian Covington Ira Globus-Harrus

April 12, 2023

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

Proves soundness of `make_quantile_score_candidates` in `mod.rs` at commit f5bb719 (outdated[1]). `make_quantile_score_candidates` returns a Transformation that takes a numeric vector database and a vector of numeric quantile candidates, and returns a vector of scores, where higher scores correspond to more accurate candidates.

Vetting History

- Pull Request #456

1 Intuition

The quantile score function scores each c in a set of candidates C.

$$s_i = -|(1-\alpha) \cdot \#(X < C_i) - \alpha \cdot \#(X > C_i)| \tag{1}$$

Where $\#(X < C_i) = |\{x \in X | x < C_i\}|$ is the number of values in X less than C_i, and similarly for other variations of inequalities. The scalar score function can be equivalently stated:

$$\begin{aligned} s_i &= -|(1-\alpha) \cdot \#(X < c) - \alpha \cdot \#(X > c)| && (2)\\ &= -|(1-\alpha) \cdot \#(X < c) - \alpha \cdot (|X| - \#(X < c) - \#(X = c))| && (3)\\ &= -|\#(X < c) - \alpha \cdot (|X| - \#(X = c))| && (4) \end{aligned}$$

It has an intuitive interpretation as $-|candidate\_rank - ideal\_rank|$, where the absolute distance between the candidate and ideal rank penalizes the score. The ideal rank does not include values in the dataset equal to the candidate. This scoring function considers higher scores better, and the score is maximized at zero when the candidate rank is equivalent to the rank at the ideal α-quantile.

The scalar scorer is almost equivalent to Smith's[1], but adjusts for a source of bias when there are values in the dataset equal to the candidate. For comparison, we can equivalently write the OpenDP scorer as if there were some α-discount on dataset entries equal to the candidate.

$$\begin{aligned} \textit{OpenDP} \quad & -|\#(X < c) + \alpha \cdot \#(X = c) - \alpha \cdot |X|| \\ \textit{Smith} \quad & -|\#(X < c) + 1 \cdot \#(X = c) - \alpha \cdot |X|| \end{aligned}$$

Observing that $\#(X \leq c) = \#(X < c) + 1 \cdot \#(X = c)$.

[1] See new changes with `git diff f5bb719..5fe96270 rust/src/transformations/quantile_score_candidates/mod.rs`

1.1 Examples

Let $X = \{0, 1, 2, 3, 4\}$ and $\alpha = 0.5$ (median):

$$\begin{aligned}
score(X, 0, \alpha) &= -|0 - .5 \cdot (5 - 1)| = -2 \\
score(X, 1, \alpha) &= -|1 - .5 \cdot (5 - 1)| = -1 \\
score(X, 2, \alpha) &= -|2 - .5 \cdot (5 - 1)| = -0 \\
score(X, 3, \alpha) &= -|3 - .5 \cdot (5 - 1)| = -1 \\
score(X, 4, \alpha) &= -|4 - .5 \cdot (5 - 1)| = -2
\end{aligned}$$

The score is maximized by the candidate at the true median.
Let $X = \{0, 1, 2, 3, 4, 5\}$ and $\alpha = 0.5$ (median):

$$\begin{aligned}
score(X, 0, \alpha) &= -|0 - .5 \cdot (6 - 1)| = -2.5 \\
score(X, 1, \alpha) &= -|1 - .5 \cdot (6 - 1)| = -1.5 \\
score(X, 2, \alpha) &= -|2 - .5 \cdot (6 - 1)| = -0.5 \\
score(X, 3, \alpha) &= -|3 - .5 \cdot (6 - 1)| = -0.5 \\
score(X, 4, \alpha) &= -|4 - .5 \cdot (6 - 1)| = -1.5 \\
score(X, 5, \alpha) &= -|5 - .5 \cdot (6 - 1)| = -2.5
\end{aligned}$$

The two candidates nearest the median are scored equally and highest.
Let $X = \{0, 1, 2, 3, 4\}$ and $\alpha = 0.25$ (first quartile):

$$\begin{aligned}
score(X, 0, \alpha) &= -|0 - .25 \cdot (5 - 1)| = -1 \\
score(X, 1, \alpha) &= -|1 - .25 \cdot (5 - 1)| = -0 \\
score(X, 2, \alpha) &= -|2 - .25 \cdot (5 - 1)| = -1 \\
score(X, 3, \alpha) &= -|3 - .25 \cdot (5 - 1)| = -2 \\
score(X, 4, \alpha) &= -|4 - .25 \cdot (5 - 1)| = -3
\end{aligned}$$

As expected, the score is maximized when $c = 1$.
Let $X = \{0, 1, 2, 3, 4, 5\}$ and $\alpha = 0.25$ (first quartile):

$$\begin{aligned}
score(X, 0, \alpha) &= -|0 - .25 \cdot (6 - 1)| = -1.25 \\
score(X, 1, \alpha) &= -|1 - .25 \cdot (6 - 1)| = -0.25 \\
score(X, 2, \alpha) &= -|2 - .25 \cdot (6 - 1)| = -0.75 \\
score(X, 3, \alpha) &= -|3 - .25 \cdot (6 - 1)| = -1.75 \\
score(X, 4, \alpha) &= -|4 - .25 \cdot (6 - 1)| = -2.75 \\
score(X, 5, \alpha) &= -|5 - .25 \cdot (6 - 1)| = -3.75
\end{aligned}$$

The ideal rank is 1.25. The nearest candidate, 1, has the greatest score, followed by 2, and then 0.

2 Finite Data Types

The previous equation assumes the existence of real numbers to represent α. We instead assume α is rational, such that $\alpha = \frac{\alpha_{num}}{\alpha_{den}}$. Multiply the equation through by α_{den} to get the following, which only uses integers:

$$\text{score}(X, c, \alpha_{num}, \alpha_{den}) = -|\alpha_{den} \cdot \#(X < c) - \alpha_{num} \cdot (|X| - \#(X = c))| \tag{5}$$

This adjustment also increases the sensitivity by a factor α_{den}, but does not affect the utility. We now make the scoring strictly non-negative.

- Drop the negation and instead configure the exponential mechanism to minimize the score.
- Compute the absolute difference in a function that swaps the order of arguments to keep the sign positive.

$$\text{score}(X, c, \alpha_{num}, \alpha_{den}) = \text{abs\_diff}(\alpha_{den} \cdot \#(X < c), \alpha_{num} \cdot (|X| - \#(X = c))) \tag{6}$$

To prevent a numerical overflow when computing the arguments, first add a limit on the dataset size, l. α_{den} is chosen sufficiently small such that $\alpha_{den} \cdot l$ is representable. Instead of sampling the dataset size down, limit both counts to be at most l and replace $|X|$ with l.

$$\text{score}(X, c, \alpha_{num}, \alpha_{den}, l) = \text{abs\_diff}(\alpha_{den} \cdot \min(\#(X < c), l), \alpha_{num} \cdot \min(|X| - \#(X = c), l)) \tag{7}$$

Should we compute counts with a 64-bit integer, we might choose α_{den} to be 10,000. This would allow for a fine fractional approximation of alpha, while still leaving enough room for datasets on the order of 10^{15} elements.

3 Hoare Triple

Precondition

- `TIA` (input atom type) is a type with trait `Number`.
- `TOA` (output atom type) is a type with trait `Float`.

Function

```
def make_quantile_score_candidates(candidates: List[TIA], alpha: TOA):

    for i in range(len(candidates) - 1):
        assert candidates[i] < candidates[i + 1]

    alpha_numer, alpha_denom = alpha.into_frac(size=None)
    if alpha_numer > alpha_denom or alpha_denom == 0:
        raise ValueError("alpha must be within [0, 1]")

    # ensures that the function will not overflow
    size_limit = size * alpha_denom

    abs_dist_const = max(alpha_numer, alpha_denom.inf_sub(alpha_numer))
    sup_dist_const = abs_dist_const.inf_mul(2)

    def function(arg: List[TIA]):
        return score(arg, candidates, alpha_numer, alpha_denom, size_limit)

    def stability_map(d_in: u32):
        return TOA.inf_cast(d_in).alerting_mul(sup_dist_const)

    return Transformation(
        input_domain=VectorDomain(AtomDomain(TIA)),
```

```
        output_domain=VectorDomain(AtomDomain(TOA)),
        function=function,
        input_metric=SymmetricDistance(),
        output_metric=InfDifferenceDistance(TOA),
        stability_map=stability_map,
    )
```

Postcondition

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

1. (Appropriate output domain). For every element v in `input_domain`, `function`(v) is in `output_domain` or raises a data-independent runtime exception.
2. (Domain-metric compatibility). The domain `input_domain` matches one of the possible domains listed in the definition of `input_metric`, and likewise `output_domain` matches one of the possible domains listed in the definition of `output_metric`.
3. (Stability 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_metric`, if u, v are `d_in`-close under `input_metric` and `stability_map`(`d_in`) $\leq$ `d_out`, then `function`(u), `function`(v) are `d_out`-close under `output_metric`.

4 Proof

4.1 Appropriate Output Domain

The raw type and domain are equivalent, save for potential nullity in the atomic type. The scalar scorer structurally cannot emit null, because the input argument is non-null.

4.2 Domain-metric compatibility

On the input side, `SymmetricDistance` is well-defined on `VectorDomain`. On the output side, `InfDifferenceDistance` is well-defined on `VectorDomain` consisting of numeric elements.

4.3 Stability Guarantee

Lemma 4.1. If $d_{Sym}(X, X') = 1$, then $d_\infty(\texttt{function}(X), \texttt{function}(X')) \leq \max(\alpha_{num}, \alpha_{den} - \alpha_{num})$.

Proof. Assume $d_{Sym}(X, X') = 1$. For convenience, let $s = \texttt{function}(X)$.

$$
\begin{aligned}
d_\infty(s_i, s_i') &= \max_i |s_i - s_i'| && \text{by definition of } d_\infty \\
&= \max_i |\text{abs\_diff}(\alpha_{den} \cdot \min(\#(X < C_i), l), \alpha_{num} \cdot \min(|X| - \#(X = C_i), l)) && \text{by definition of } \texttt{function} \\
&\quad \text{abs\_diff}(\alpha_{den} \cdot \min(\#(X' < C_i), l), \alpha_{num} \cdot \min(|X'| - \#(X' = C_i), l))| \\
&= \alpha_{den} \cdot \max_i || \min(\#(X < C_i), l) - \alpha \cdot \min(|X| - \#(X = C_i), l)| \\
&\quad | \min(\#(X' < C_i), l) - \alpha \cdot \min(|X'| - \#(X' = C_i), l)|| \\
&\leq \alpha_{den} \cdot \max_i ||\#(X < C_i) - \alpha \cdot (|X| - \#(X = C_i))| \\
&\quad |\#(X' < C_i) - \alpha \cdot (|X| - \#(X' = C_i))||
\end{aligned}
$$

Consider each of the three cases of adding or removing an element in X.

Case 1. Assume X' is equal to X, but with some $X_j < C_i$ added or removed.

$$
\begin{aligned}
&= \alpha_{den} \cdot \max_i ||(1-\alpha) \cdot \#(X < C_i) - \alpha \cdot \#(X > C_i)| \\
&\qquad - |(1-\alpha) \cdot (\#(X < C_i) \pm 1) - \alpha \cdot \#(X > C_i)|| \\
&\leq \alpha_{den} \cdot \max_i ||(1-\alpha) \cdot \#(X < C_i) - \alpha \cdot \#(X > C_i)| \\
&\qquad - (|(1-\alpha) \cdot \#(X < C_i) - \alpha \cdot \#(X > C_i)| + |\pm(1-\alpha)|)| && \text{by triangle inequality} \\
&= \alpha_{den} \cdot \max_i |1-\alpha| && \text{scores cancel} \\
&= \alpha_{den} - \alpha_{num} && \text{since } \alpha \leq 1
\end{aligned}
$$

Case 2. Assume X' is equal to X, but with some $X_j > C_i$ added or removed.

$$
\begin{aligned}
&= \alpha_{den} \cdot \max_i ||(1-\alpha) \cdot \#(X < C_i) - \alpha \cdot \#(X > C_i)| \\
&\qquad - |(1-\alpha) \cdot \#(X < C_i) - \alpha \cdot (\#(X > C_i) \pm 1)|| \\
&\leq \alpha_{den} \cdot \max_i ||(1-\alpha) \cdot \#(X < C_i) - \alpha \cdot \#(X > C_i)| \\
&\qquad - (|(1-\alpha) \cdot \#(X < C_i) - \alpha \cdot \#(X > C_i)| + |\pm\alpha|)| && \text{by triangle inequality} \\
&= \alpha_{den} \cdot \max_i |\alpha| && \text{scores cancel} \\
&= \alpha_{num} && \text{since } \alpha \geq 0
\end{aligned}
$$

Case 3. Assume X' is equal to X, but with some $X_j = C_i$ added or removed.

$$
\begin{aligned}
&= \alpha_{den} \cdot \max_i ||(1-\alpha) \cdot \#(X < C_i) - \alpha \cdot \#(X > C_i)| \\
&\qquad - |(1-\alpha) \cdot \#(X < C_i) - \alpha \cdot \#(X > C_i)|| \\
&= 0 && \text{no change in score}
\end{aligned}
$$

Take the union bound over all cases.

$$\max_{X \sim X'} d_\infty(s_i, s_i') \leq \max(\alpha_{num}, \alpha_{den} - \alpha_{num})$$

□

Take any two elements X, X' in the `input_domain` and any pair (`d_in`, `d_out`), where `d_in` has the associated type for `input_metric` and `d_out` has the associated type for `output_metric`. Assume X, X' are `d_in`-close under `input_metric` and that `stability_map(d_in)` $\leq$ `d_out`.

$$
\begin{aligned}
\texttt{d\_out} &= \max_{X \sim X'} d_{IDD}(s, s') && \text{where } s = \texttt{function}(X) \\
&= \max_{X \sim X'} \max_{ij} |(s_i - s'_i) - (s_j - s'_j)| && \text{by definition of InfDifferenceDistance} \\
&\leq 2 \max_{X \sim X'} \max_{i} |s_i - s'_i| && s \text{ is not monotonic; take looser bound} \\
&\leq 2 \sum_{j}^{\texttt{d\_in}} \max_{Z_j \sim Z_{j+1}} \max_{i} |s_{i,j} - s_{i,j+1}| && \text{by path property where } d_{Sym}(Z_i, Z_{i+1}) = 1, X = Z_0 \text{ and } Z_{\texttt{d\_in}} = X' \\
&\leq 2 \sum_{j}^{\texttt{d\_in}} \max(\alpha_{num}, \alpha_{den} - \alpha_{num}) && \text{by 4.1} \\
&\leq 2 \cdot \texttt{d\_in} \cdot \max(\alpha_{num}, \alpha_{den} - \alpha_{num})
\end{aligned}
$$

It is shown that `function`(X), `function`(X') are `d_out`-close under `output_metric`.

References

[1] Adam Smith. Privacy-preserving statistical estimation with optimal convergence rates. In *Proceedings of the Forty-Third Annual ACM Symposium on Theory of Computing*, STOC '11, page 813–822, New York, NY, USA, 2011. Association for Computing Machinery.