fn make_quantile_score_candidates

Michael Shoemate Christian Covington Ira Globus-Harrus

April 1, 2025

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Proves soundness of `make_quantile_score_candidates` in `mod.rs` at commit f5bb719 (outdated[1]).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.

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

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.

$$\begin{array}{cl} OpenDP & -|\#(x < c) + \alpha \cdot \#(x = c) - \alpha \cdot |x|| \\ Smith & -|\#(x < c) + 1 \cdot \#(x = c) - \alpha \cdot |x|| \end{array}$$

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

[1] See new changes with `git diff f5bb719..5da95cf 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:

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

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.

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

To prevent a numerical overflow when computing the arguments to `abs_diff`, first choose a data type that the scores are to be represented in. If the number of records is greater than can be represented in this data type, then sample the dataset down to at most this number of records. Notice that when any given record is added or removed, the counts differ by no more than they would have without this sampling down. In the OpenDP implementation, the dataset size may be no greater than the max value of a Rust usize, because each index into the dataset maps to a distinct computer memory address.

Now allocate some of the bits of the data type for the alpha denominator, and use the remaining bits for counts of up to l, where l is the effective dataset size. From this set-up, we choose an α_{den} such that $\alpha_{den} \cdot l$ is representable. Since $\alpha_{num} \leq \alpha_{den}$, $\alpha_{num} \cdot l$ is representable. Since the dataset size fits in the choice of data type, then $|x|$ is representable. Therefore, no quantity in the following equation is not representable.

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

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`.
- `A` (alpha type) is a type with trait `Float`.
- `MI` is a type with trait `ARDatasetMetric`.

Function

```python
def make_quantile_score_candidates(
    input_domain: VectorDomain[AtomDomain[TIA]],
    input_metric: MI,
    candidates: list[TIA],
    alpha: A
) -> Transformation:

    input_domain.element_domain.assert_non_null()

    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:
```

```python
        raise ValueError("alpha must be within [0, 1]")

    if input_domain.size is not None:
        # to ensure that the function will not overflow
        input_domain.size.inf_mul(alpha_denom)
        size_limit = input_domain.size
    else:
        size_limit = (usize.MAX).neg_inf_div(alpha_den)

    def function(arg: list[TIA]) -> list[usize]:
        return compute_score(arg, candidates, alpha_numer, alpha_denom, size_limit)

    if input_domain.size is not None:
        def stability_map(d_in: u32) -> usize:
            return TOA.inf_cast(d_in // 2).inf_mul(2).inf_mul(alpha_denom)
    else:
        abs_dist_const: usize = max(alpha_numer, alpha_denom.inf_sub(alpha_numer))
        stability_map = new_stability_map_from_constant(abs_dist_const, QO=usize)

    return Transformation(
        input_domain=input_domain,
        output_domain=VectorDomain(
            element_domain=AtomDomain(T=usize),
            size=len(candidates)),
        function=function,
        input_metric=input_metric,
        output_metric=LInfDistance(Q=usize),
        stability_map=stability_map,
    )
```

Postcondition

Theorem 3.1. For every setting of the input parameters (`input_domain, input_metric, candidates, alpha, TIA, A, MI`) 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 x in `input_domain`, `function`(x) is in `output_domain` or raises a data-independent runtime exception.

2. (Stability guarantee). For every pair of elements x, 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_metric`, if x, x' are `d_in`-close under `input_metric`, `stability_map`(`d_in`) does not raise an exception, and `stability_map`(`d_in`) $\leq$ `d_out`, then `function`(x), `function`(x') 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. Therefore the output of the function is a member of the output domain.

4.2 Stability Guarantee

The constructor first performs checks to ensure that the preconditions on `compute_score` are met. It checks that vectors in the input domain do not contain null values, that the candidates are strictly increasing, that alpha is fractional and in the range $[0, 1]$, and computes a `size_limit` for which `size_limit` $\cdot$ `alpha_den` does not overflow a `usize`. Thus by the definition of `compute_score`, for each candidate, the response from the function is:

$$\text{compute\_score}(x, c, \alpha_{num}, \alpha_{den}, l) = |\alpha_{den} \cdot \min(\#(x < c), l), \alpha_{num} \cdot \min(|x| - \#(x = c), l)| \quad (8)$$

The sensitivity of this function differs depending on if the size of the input vector is known.

4.2.1 Unknown Size Stability

First, consider the case where the size is unknown.

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

Proof. Assume $d_{Sym}(x, x') = 1$.

$$\begin{aligned}
&d_\infty(\texttt{function}(x)_i, \texttt{function}(x')_i) \\
&= \max_i |\texttt{function}(x)_i - \texttt{function}(x')_i| && \text{by definition of } d_\infty \\
&= \max_i |\text{abs_diff}(\alpha_{den} \cdot \min(\#(x < C_i), l), \alpha_{num} \cdot \min(|x| - \#(x = C_i), l)) && \text{by definition of } \texttt{function} \\
&\qquad \text{abs_diff}(\alpha_{den} \cdot \min(\#(x' < C_i), l), \alpha_{num} \cdot \min(|x'| - \#(x' = C_i), l))| \\
&= \alpha_{den} \cdot \max_i || \min(\#(x < C_i), l) - \alpha \cdot \min(|x| - \#(x = C_i), l)| \\
&\qquad | \min(\#(x' < C_i), l) - \alpha \cdot \min(|x'| - \#(x' = C_i), l)|| \\
&\le \alpha_{den} \cdot \max_i ||\#(x < C_i) - \alpha \cdot (|x| - \#(x = C_i))| \\
&\qquad |\#(x' < C_i) - \alpha \cdot (|x| - \#(x' = C_i))||
\end{aligned}$$

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.

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

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

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

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.

$$\leq \max(\alpha_{num}, \alpha_{den} - \alpha_{num})$$

□

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`.

$$
\begin{aligned}
\texttt{d\_out} &= \max_{x \sim x'} d_\infty(s, s') && \text{where } s = \texttt{function}(x) \\
&= \max_{x \sim x'} \max_i |s_i - s'_i| && \text{by definition of } \texttt{LInfDistance}\text{, without monotonicity} \\
&\leq \sum_j^{\texttt{d\_in}} \max_{Z_j \sim Z_{j+1}} \max_i |s_{i,j} - s_{i,j+1}| && \text{by path property } d_{Sym}(Z_i, Z_{i+1}) = 1, x = Z_0 \text{ and } x' = Z_{\texttt{d\_in}} \\
&\leq \sum_j^{\texttt{d\_in}} \max(\alpha_{num}, \alpha_{den} - \alpha_{num}) && \text{by 4.1} \\
&\leq \texttt{d\_in} \cdot \max(\alpha_{num}, \alpha_{den} - \alpha_{num})
\end{aligned}
$$

This formula matches the stability map in the case where the dataset size is unknown.

4.2.2 Known Size Stability

Now consider the case where the dataset size is known.

Lemma 4.2. If $d_{CO}(x, x') \leq 1$, then $d_\infty(\texttt{function}(x), \texttt{function}(x')) \leq 2 \cdot \alpha_{den}$.

Proof. Assume $d_{CO}(x, x') \leq 1$.

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

Consider each of the four cases of changing a row in x.

Case 1. Assume x' is equal to x, but with some $x_j < C_i$ replaced with $x'_j > C_i$.

$$
\begin{aligned}
&= 2 \cdot \alpha_{den} \cdot \max_i ||(1-\alpha) \cdot \#(x < C_i) - \alpha \cdot \#(x > C_i)| \\
&\qquad - (1-\alpha) \cdot (\#(x < C_i) - 1) - \alpha \cdot (\#(x > C_i) + 1)|| && \text{by definition of } \texttt{function} \\
&\le 2 \cdot \alpha_{den} \cdot \max_i ||(1-\alpha) \cdot \#(x < C_i) - \alpha \cdot \#(x > C_i)| \\
&\qquad - (|(1-\alpha) \cdot \#(x < C_i) - \alpha \cdot \#(x > C_i)| + |1|)| && \text{by triangle inequality} \\
&= 2 \cdot \alpha_{den} \cdot \max_i |1| && \text{scores cancel} \\
&= 2 \cdot \alpha_{den}
\end{aligned}
$$

Case 2. Assume x' is equal to x, but with some $x_j > C_i$ replaced with $x'_j < C_i$.

$$= 2 \cdot \alpha_{den}$$

by symmetry, follows from Case 1.

Case 3. Assume x' is equal to x, but with some $x_j \neq C_i$ replaced with C_i.

$$\le 2 \cdot \max(\alpha_{num}, \alpha_{den} - \alpha_{num})$$

equivalent to one removal (see `make_quantile_score_candidates`)

Case 4. Assume x' is equal to x, but with some $x_j = C_i$ replaced with $x'_j \neq C_i$.

$$\le 2 \cdot \max(\alpha_{num}, \alpha_{den} - \alpha_{num})$$

equivalent to one addition (see `make_quantile_score_candidates`)

Take the union bound over all cases.

$$d_\infty(s_i, s'_i) \le \max(2 \cdot \alpha_{den}, 2 \cdot \max(\alpha_{num}, \alpha_{den} - \alpha_{num})) = 2 \cdot \alpha_{den}$$

since $\max(\alpha, 1-\alpha) \le 1$

□

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)` $\le$ `d_out`.

$$
\begin{aligned}
\texttt{d\_out} &= \max_{x \sim x'} d_{\infty}(s, s') \\
&= \max_{x \sim x'} \max_i |s_i - s'_i| && \text{by definition of } \texttt{LInfDistance}\text{, without monotonicity} \\
&\le \sum_{j}^{\texttt{d\_in}//2} \max_{Z_j \sim Z_{j+1}} \max_i |s_{i,j} - s_{i,j+1}| && \text{by path property } d_{CO}(Z_i, Z_{i+1}) = 1, x = Z_0 \text{ and } Z_{\texttt{d\_in}} = x' \\
&\le \sum_{j}^{\texttt{d\_in}//2} 2 \cdot \alpha_{den} && \text{by 4.2} \\
&\le 2 \cdot (\texttt{d\_in}//2) \cdot \alpha_{den}
\end{aligned}
$$

This formula matches the stability map in the case where the dataset size is known.

4.2.3 Conclusion

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)` $\le$ `d_out`.

By 4.2.1 and 4.2.2 it is shown that `function`(x), `function`(x') are `d_out`-close under `output_metric` for any choice of input arguments.

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.