fn make_impute_uniform_float

Noah Chung

Proves soundness of make_impute_uniform_float.

make_impute_constant returns a Transformation that replaces NaN values in 'Vec<TA>' with uniformly distributed floats within 'bounds.'

1 Hoare Triple

Precondition

- Type TA (atomic input type) must have trait Float and traint bound SampleUniform. Float implies TA implies that it can be sampled from SampleUniform and has the trait bound:
  - CheckNull so that TA is a valid atomic type for AtomDomain
- M (metric) is a type with trait DatasetMetric. DatasetMetric is used to restrict the set of valid metrics to those which measure distances between datasets.
- MetricSpace is implemented for (VectorDomain<AtomDomain<TA», M)

Pseudocode

```python

def make_impute_uniform_float(
    input_domain: VectorDomain[AtomDomain[TA]], #
    input_metric: M, #
    bounds: (TA, TA)
):
    output_domain = AtomDomain(TA) #
    output_metric = M #

    let lower, upper = bounds #
    if lower.isnan():
        raise MakeTransformation("lower may not be nan")
    if upper.isnan():
        raise MakeTransformation("upper may not be nan")
    if lower >= upper:
        raise MakeTransformation("lower must be smaller than upper")

    result = make_row_by_row(
        input_domain,
        input_metric,
        output_domain,  # Using output_domain instead of atom_domain which wasn't defined
        lambda v: v if not is_null(v) else (
            (lambda: (rng := GeneratorOpenDP()).uniform(lower, upper)
             if not rng.error() else lower)()
        )
    )
    return result #
```

Postcondition

Theorem 1.1. For every setting of the input parameters (input_domain, input_metric, bounds) to make_impute_uniform_float such that the given preconditions hold, make_impute_uniform_float 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.

2 Proofs

Proof. **(Part 1 – appropriate output domain).** The output_domain is AtomDomain(TA), so it is sufficient to show that function returns a dataset of non-null values of type TA. We rely on the correctness of the function make_row_by_row. The function make_row_by_row returns a dataset in input_domain.translate(output_row_domain), if row_function is a mapping between input_domain's row domain to output_row_domain. This is satisfied by the precondition on input_domain. Thus, in all cases, the function (from line 27) returns a non-null dataset of type TA. □

Proof. **(Part 2 – stability map).** Take any two elements u, v 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 u, v are d_in-close under input_metric and that stability_map(d_in) $\leq$ d_out. These assumptions are used to establish the following inequality:

$$\begin{aligned} d_M(\texttt{function}(u), \texttt{function}(v)) &= d_M([f(u_1), f(u_2), ...], [f(v_1), f(v_2), ...]) && \text{since } \texttt{DO} \text{ is a } \texttt{DatasetDomain} \\ &\leq \texttt{d\_out} && \text{by } \texttt{make\_row\_by\_row} \text{ correctness} \end{aligned}$$

As the input data is not modified before being input into make_row_by_row, by correctness of make_row_by_row, it is shown that function(u), function(v) are d_out-close under output_metric. □