

fn make_vec

Michael Shoemate

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

Proves soundness of the implementation of `make_vec` in `mod.rs` at commit `f5bb719` (outdated¹).

This transformation simply wraps an input scalar in a singleton vec. The output metric then becomes an Lp distance.

1 Hoare Triple

Precondition

Compiler-Verified

- Generic T implements trait `Number`
- Generic Q implements trait `Number`

User-Verified

None

Pseudocode

```
1 def make_vec(  
2     input_space: tuple[AtomDomain[T], AbsoluteDistance[Q]],  
3 ) -> Transformation[  
4     AtomDomain[T], VectorDomain[AtomDomain[T]], AbsoluteDistance[Q], LpDistance[P, Q]  
5 ]:  
6     input_domain, input_metric = input_space  
7     return Transformation.new(  
8         input_domain,  
9         VectorDomain.new(input_domain).with_size(1),  
10        lambda arg: [arg],  
11        input_metric,  
12        LpDistance.default(),  
13        lambda d_in: d_in,  
14    )
```

Postcondition

Theorem 1.1.

Theorem 1.2. For every setting of the input parameters (`input_space`, `T`, `Q`) to `make_vec` such that the given preconditions hold, `make_vec` raises an error (at compile time or run time) or returns a valid transformation. A valid transformation has the following properties:

¹See new changes with `git diff f5bb719..5907d96 rust/src/transformations/scalar_to_vector/mod.rs`

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. (Appropriate output domain). For every member x in `input_domain`, `function(x)` is in `output_domain` or raises a data-independent runtime error.
3. (Stability guarantee). For every pair of members x and x' in `input_domain` and for every pair $(\mathbf{d_in}, \mathbf{d_out})$, where $\mathbf{d_in}$ has the associated type for `input_metric` and $\mathbf{d_out}$ has the associated type for `output_metric`, if x, x' are $\mathbf{d_in}$ -close under `input_metric`, `stability_map(d_in)` does not raise an error, and `stability_map(d_in) = d_out`, then `function(x), function(x')` are $\mathbf{d_out}$ -close under `output_metric`.

Proof. The function is infallible, and the output domain trivially follows, since all output vectors are length-one. For all x in the input domain, the output of `make_vec` is a vector of length 1, so the output domain is trivially valid. The function is 1-stable because:

$$\begin{aligned}
& \mathbf{d_out} & (1) \\
& = \max_{x \sim x'} d_{Lp}(f(x), f(x')) & (2) \\
& = \max_{x \sim x'} \left(\sum_i (x_i - x'_i)^p \right)^{1/p} & (3) \\
& = \max_{x \sim x'} ((x_1 - x'_1)^p)^{1/p} & (4) \\
& = \max_{x \sim x'} |x_1 - x'_1| & (5) \\
& = \max_{x \sim x'} d_{Abs}(x_1, x'_1) & (6) \\
& = 1 \cdot \mathbf{d_in} & (7) \\
& & (8)
\end{aligned}$$

For every pair of elements x, x' in `input_domain` and for every pair $(\mathbf{d_in}, \mathbf{d_out})$, where $\mathbf{d_in}$ has the associated type for `input_metric` and $\mathbf{d_out}$ has the associated type for `output_metric`, if x, x' are $\mathbf{d_in}$ -close under `input_metric`, `stability_map(d_in)` does not raise an exception, and `stability_map(d_in) ≤ d_out`, then `function(x), function(x')` are $\mathbf{d_out}$ -close under `output_metric`. \square