tf.contrib.distributions.bijectors.AbsoluteValue

Contents

Class `AbsoluteValue`

Inherits From: `Bijector`

Defined in `tensorflow/contrib/distributions/python/ops/bijectors/absolute_value_impl.py`.

Computes `Y = g(X) = Abs(X)`, element-wise.

This non-injective bijector allows for transformations of scalar distributions with the absolute value function, which maps `(-inf, inf)` to `[0, inf)`.

- For `y in (0, inf)`, `AbsoluteValue.inverse(y)` returns the set inverse `{x in (-inf, inf) : |x| = y}` as a tuple, `-y, y`.
- `AbsoluteValue.inverse(0)` returns `0, 0`, which is not the set inverse (the set inverse is the singleton `{0}`), but "works" in conjunction with `TransformedDistribution` to produce a left semi-continuous pdf.
- For `y < 0`, `AbsoluteValue.inverse(y)` happily returns the wrong thing, `-y, y`. This is done for efficiency. If `validate_args == True`, `y < 0` will raise an exception.

```
abs = ds.bijectors.AbsoluteValue()

abs.forward([-1., 0., 1.])
==> [1., 0.,  1.]

abs.inverse(1.)
==> [-1., 1.]

# The |dX/dY| is constant, == 1.  So Log|dX/dY| == 0.
abs.inverse_log_det_jacobian(1.)
==> [0., 0.]

# Special case handling of 0.
abs.inverse(0.)
==> [0., 0.]

abs.inverse_log_det_jacobian(0.)
==> [0., 0.]
```

Properties

`dtype`

dtype of `Tensor` s transformable by this distribution.

`event_ndims`

Returns then number of event dimensions this bijector operates on.

`graph_parents`

Returns this `Bijector` 's graph_parents as a Python list.

`is_constant_jacobian`

Returns true iff the Jacobian is not a function of x.

> **Note:** Jacobian is either constant for both forward and inverse or neither.

Returns:

- `is_constant_jacobian` : Python `bool` .

`name`

Returns the string name of this `Bijector` .

`validate_args`

Returns True if Tensor arguments will be validated.

Methods

`__init__`

```
__init__(
    event_ndims=0,
    validate_args=False,
    name='absolute_value'
)
```

Instantiates the `AbsoluteValue` bijector.

Args:

- `event_ndims` : Python scalar indicating the number of dimensions associated with a particular draw from the distribution. Currently only zero is supported.
- `validate_args` : Python `bool` indicating whether arguments should be checked for correctness, in particular whether inputs to `inverse` and `inverse_log_det_jacobian` are non-negative.
- `name` : Python `str` name given to ops managed by this object.

Raises:

- `ValueError`: If `event_ndims` is not zero.

`forward`

```
forward(
    x,
    name='forward'
)
```

Returns the forward `Bijector` evaluation, i.e., X = g(Y).

Args:

- `x`: `Tensor`. The input to the "forward" evaluation.
- `name`: The name to give this op.

Returns:

`Tensor`.

Raises:

- `TypeError`: if `self.dtype` is specified and `x.dtype` is not `self.dtype`.
- `NotImplementedError`: if `_forward` is not implemented.

`forward_event_shape`

```
forward_event_shape(input_shape)
```

Shape of a single sample from a single batch as a `TensorShape`.

Same meaning as `forward_event_shape_tensor`. May be only partially defined.

Args:

- `input_shape`: `TensorShape` indicating event-portion shape passed into `forward` function.

Returns:

- `forward_event_shape_tensor`: `TensorShape` indicating event-portion shape after applying `forward`. Possibly unknown.

`forward_event_shape_tensor`

```
forward_event_shape_tensor(
    input_shape,
    name='forward_event_shape_tensor'
)
```

Shape of a single sample from a single batch as an `int32` 1D `Tensor`.

Args:

- `input_shape`: `Tensor`, `int32` vector indicating event-portion shape passed into `forward` function.
- `name`: name to give to the op

Returns:

- `forward_event_shape_tensor`: `Tensor`, `int32` vector indicating event-portion shape after applying `forward`.

`forward_log_det_jacobian`

```
forward_log_det_jacobian(
    x,
    name='forward_log_det_jacobian'
)
```

Returns both the forward_log_det_jacobian.

Args:

- `x`: `Tensor`. The input to the "forward" Jacobian evaluation.
- `name`: The name to give this op.

Returns:

`Tensor`, if this bijector is injective. If not injective this is not implemented.

Raises:

- `TypeError`: if `self.dtype` is specified and `y.dtype` is not `self.dtype`.
- `NotImplementedError`: if neither `_forward_log_det_jacobian` nor {`_inverse`, `_inverse_log_det_jacobian`} are implemented, or this is a non-injective bijector.

`inverse`

```
inverse(
    y,
    name='inverse'
)
```

Returns the inverse `Bijector` evaluation, i.e., X = g^{-1}(Y).

Args:

- `y`: `Tensor`. The input to the "inverse" evaluation.
- `name`: The name to give this op.

Returns:

`Tensor`, if this bijector is injective. If not injective, returns the k-tuple containing the unique `k` points `(x1, ..., xk)` such that `g(xi) = y`.

Raises:

- `TypeError`: if `self.dtype` is specified and `y.dtype` is not `self.dtype`.
- `NotImplementedError`: if `_inverse` is not implemented.

`inverse_event_shape`

```
inverse_event_shape(output_shape)
```

Shape of a single sample from a single batch as a `TensorShape`.

Same meaning as `inverse_event_shape_tensor`. May be only partially defined.

Args:

- `output_shape`: `TensorShape` indicating event-portion shape passed into `inverse` function.

Returns:

- `inverse_event_shape_tensor`: `TensorShape` indicating event-portion shape after applying `inverse`. Possibly unknown.

`inverse_event_shape_tensor`

```
inverse_event_shape_tensor(
    output_shape,
    name='inverse_event_shape_tensor'
)
```

Shape of a single sample from a single batch as an `int32` 1D `Tensor`.

Args:

- `output_shape`: `Tensor`, `int32` vector indicating event-portion shape passed into `inverse` function.
- `name`: name to give to the op

Returns:

- `inverse_event_shape_tensor`: `Tensor`, `int32` vector indicating event-portion shape after applying `inverse`.

`inverse_log_det_jacobian`

```
inverse_log_det_jacobian(
    y,
    name='inverse_log_det_jacobian'
)
```

Returns the (log o det o Jacobian o inverse)(y).

Mathematically, returns: `log(det(dX/dY))(Y)`. (Recall that: `X=g^{-1}(Y)`.)

Note that `forward_log_det_jacobian` is the negative of this function, evaluated at `g^{-1}(y)`.

Args:

- `y`: `Tensor`. The input to the "inverse" Jacobian evaluation.
- `name`: The name to give this op.

Returns:

`Tensor`, if this bijector is injective. If not injective, returns the tuple of local log det Jacobians, `log(det(Dg_i^{-1}(y)))`, where `g_i` is the restriction of `g` to the `ith` partition `Di`.

Raises:

- `TypeError`: if `self.dtype` is specified and `y.dtype` is not `self.dtype`.
- `NotImplementedError`: if `_inverse_log_det_jacobian` is not implemented.

Last updated November 2, 2017.