

tf.contrib.distributions.bijectors.Inline

Contents

Class Inline

Properties

dtype

event_ndims

Class **Inline**Inherits From: **Bijector**Defined in [tensorflow/contrib/distributions/python/ops/bijectors/inline_impl.py](#).See the guide: [Random variable transformations \(contrib\) > Bijectors](#)

Bijector constructed from custom callables.

Example Use:

```
exp = Inline(
    forward_fn=tf.exp,
    inverse_fn=tf.log,
    inverse_log_det_jacobian_fn=(
        lambda y: -tf.reduce_sum(tf.log(y), axis=-1)),
    name="exp")
```

The above example is equivalent to the **Bijector** `Exp(event_ndims=1)`.

Properties

dtypedtype of **Tensor**s transformable by this distribution.**event_ndims**

Returns then number of event dimensions this bijector operates on.

graph_parentsReturns 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__(
    forward_fn=None,
    inverse_fn=None,
    inverse_log_det_jacobian_fn=None,
    forward_log_det_jacobian_fn=None,
    forward_event_shape_fn=None,
    forward_event_shape_tensor_fn=None,
    inverse_event_shape_fn=None,
    inverse_event_shape_tensor_fn=None,
    is_constant_jacobian=False,
    validate_args=False,
    name='inline'
)
```

Creates a `Bijector` from callables.

Args:

- `forward_fn`: Python callable implementing the forward transformation.
- `inverse_fn`: Python callable implementing the inverse transformation.
- `inverse_log_det_jacobian_fn`: Python callable implementing the log o det o jacobian of the inverse transformation.
- `forward_log_det_jacobian_fn`: Python callable implementing the log o det o jacobian of the forward transformation.
- `forward_event_shape_fn`: Python callable implementing non-identical static event shape changes. Default: shape is assumed unchanged.
- `forward_event_shape_tensor_fn`: Python callable implementing non-identical event shape changes. Default: shape is assumed unchanged.
- `inverse_event_shape_fn`: Python callable implementing non-identical static event shape changes. Default: shape is assumed unchanged.
- `inverse_event_shape_tensor_fn`: Python callable implementing non-identical event shape changes. Default: shape is assumed unchanged.
- `is_constant_jacobian`: Python `bool` indicating that the Jacobian is constant for all input arguments.

- `validate_args`: Python `bool` indicating whether arguments should be checked for correctness.
- `name`: Python `str`, name given to ops managed by this object.

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 $(\log \circ \det \circ \text{Jacobian} \circ \text{inverse})(y)$.

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 i th partition D_i .

Raises:

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

Except as otherwise noted, the content of this page is licensed under the [Creative Commons Attribution 3.0 License](#), and code samples are licensed under the [Apache 2.0 License](#). For details, see our [Site Policies](#). Java is a registered trademark of Oracle and/or its affiliates.

Last updated November 2, 2017.

Stay Connected

[Blog](#)

[GitHub](#)

[Twitter](#)

Support

[Issue Tracker](#)

[Release Notes](#)

[Stack Overflow](#)

English

[Terms](#) | [Privacy](#)