tf.contrib.distributions.bijectors.Softplus

Contents

Class `Softplus`

Inherits From: `Bijector`

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

See the guide: Random variable transformations (contrib) > Bijectors

Bijector which computes `Y = g(X) = Log[1 + exp(X)]`.

The softplus `Bijector` has the following two useful properties:

- The domain is the positive real numbers
- `softplus(x) approx x`, for large `x`, so it does not overflow as easily as the `Exp` `Bijector`.

The optional nonzero `hinge_softness` parameter changes the transition at zero. With `hinge_softness = c`, the bijector is:

```
r large `x >> 1`, `c * Log[1 + exp(x / c)] approx c * Log[exp(x / c)] = x`,
 the behavior for large `x` is the same as the standard softplus.

 `c > 0` approaches 0 from the right, `f_c(x)` becomes less and less soft,
proaching `max(0, x)`.

`c = 1` is the default.
`c > 0` but small means `f(x) approx ReLu(x) = max(0, x)`.
`c < 0` flips sign and reflects around the `y-axis`: `f_{-c}(x) = -f_c(-x)`.
`c = 0` results in a non-bijective transformation and triggers an exception.

Example Use:
```

Create the Y=g(X)=softplus(X) transform which works only on Tensors with 1 # batch ndim and 2 event ndims (i.e., vector of matrices). softplus = Softplus(event_ndims=2) x = [[[1., 2], [3, 4]], [[5, 6], [7, 8]]] log(1 + exp(x)) == softplus.forward(x) log(exp(x) - 1) == softplus.inverse(x)

```
Note: log(.) and exp(.) are applied element-wise but the Jacobian is a
reduction over the event space.

 Properties

3 id="dtype"><code>dtype</code></h3>

ype of `Tensor`s transformable by this distribution.

3 id="event_ndims"><code>event_ndims</code></h3>

turns then number of event dimensions this bijector operates on.

3 id="graph_parents"><code>graph_parents</code></h3>

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

3 id="hinge_softness"><code>hinge_softness</code></h3>

3 id="is_constant_jacobian"><code>is_constant_jacobian</code></h3>

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

te: Jacobian is either constant for both forward and inverse or neither.

## Returns:

<b>`is_constant_jacobian`</b>: Python `bool`.

3 id="name"><code>name</code></h3>

turns the string name of this `Bijector`.

3 id="validate_args"><code>validate_args</code></h3>

turns True if Tensor arguments will be validated.

 Methods

3 id="__init__"><code>__init__</code></h3>
```

```
__init__(
    *args,
    **kwargs
)
```

kwargs:

- **`hinge_softness`**: Nonzero floating point **`Tensor`**. Controls the softness of what would otherwise be a kink at the origin. Default is 1.0

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.