tfp.bijectors.Log

View source (https://github.com/tensorflow/probability/blob/v0.13.0/tensorflow_probability/python/bijon L90) GitHub

Compute `Y = log(X)`. This is `Invert(Exp())`.

Inherits From: `Invert` (https://www.tensorflow.org/probability/api_docs/python/tfp/bijectors/Invert), `AutoCompositeTensorBijector` (https://www.tensorflow.org/probability/api_docs/python/tfp/bijectors/AutoCompositeTensorBijector)

```
tfp.bijectors.Log(
    validate_args=False, name='log'
)
```

Args

`bijector`	Bijector instance.
`validate_args`	Python `bool` indicating whether arguments should be checked for correctness.
`parameters`	Locals dict captured by subclass constructor, to be used for copy/slice re-instantiation operators.
`name`	Python `str`, name given to ops managed by this object.

Attributes

`bijector`	
`dtype`	
`forward_min_event_ndims`	Returns the minimal number of dimensions bijector.forward operates on.

	Multipart bijectors return structured `ndims`, which indicates the expected structure of their inputs. Some multipart bijectors, notably Composites, may return structures of `None`.
`graph_parents`	Returns this `Bijector`'s graph_parents as a Python list.
`has_static_min_event_ndims`	Returns True if the bijector has statically-known `min_event_ndims`. (deprecated) **Warning:** THIS FUNCTION IS DEPRECATED. It will be removed after 2021-08-01. Instructions for updating: `min_event_ndims` is now static for all bijectors; this property is no longer needed.
`inverse_min_event_ndims`	Returns the minimal number of dimensions bijector.inverse operates on. Multipart bijectors return structured `event_ndims`, which indicates the expected structure of their outputs. Some multipart bijectors, notably Composites, may return structures of `None`.
`is_constant_jacobian`	Returns true iff the Jacobian matrix is not a function of x. **Note:** Jacobian matrix is either constant for both forward and inverse or neither.
`name`	Returns the string name of this `Bijector`.
`name_scope`	Returns a `tf.name_scope` (https://www.tensorflow.org/api_docs/python/tf/name_scope) instance for this class.
`non_trainable_variables`	Sequence of non-trainable variables owned by this module and its submodules. **Note:** this method uses reflection to find variables on the current instance and submodules. For performance reasons you may wish to cache the result of calling this method if you don't expect the return value to change.
`parameters`	Dictionary of parameters used to instantiate this `Bijector`.

`submodules`	Sequence of all sub-modules.
	Submodules are modules which are properties of this module, or found as properties of modules which are properties of this module (and so on).
	
	`>>> a = tf.Module()`
	`>>> b = tf.Module()`
	`>>> c = tf.Module()`
	`>>> a.b = b`
	`>>> b.c = c`
	`>>> list(a.submodules) == [b, c]`
	`True`
	`>>> list(b.submodules) == [c]`
	`True`
	`>>> list(c.submodules) == []`
	`True`

`trainable_variables`	Sequence of trainable variables owned by this module and its submodules. **Note:** this method uses reflection to find variables on the current instance and submodules. For performance reasons you may wish to cache the result of calling this method if you don't expect the return value to change.
`validate_args`	Returns True if Tensor arguments will be validated.
`variables`	Sequence of variables owned by this module and its submodules. **Note:** this method uses reflection to find variables on the current instance and submodules. For performance reasons you may wish to cache the result of calling this method if you don't expect the return value to change.

Attributes

`bijector`	
`dtype`	
`forward_min_event_ndims`	Returns the minimal number of dimensions bijector.forward operates on.

	Multipart bijectors return structured `ndims`, which indicates the expected structure of their inputs. Some multipart bijectors, notably Composites, may return structures of `None`.
`graph_parents`	Returns this `Bijector`'s graph_parents as a Python list.
`has_static_min_event_ndims`	Returns True if the bijector has statically-known `min_event_ndims`. (deprecated) **Warning:** THIS FUNCTION IS DEPRECATED. It will be removed after 2021-08-01. Instructions for updating: `min_event_ndims` is now static for all bijectors; this property is no longer needed.
`inverse_min_event_ndims`	Returns the minimal number of dimensions bijector.inverse operates on. Multipart bijectors return structured `event_ndims`, which indicates the expected structure of their outputs. Some multipart bijectors, notably Composites, may return structures of `None`.
`is_constant_jacobian`	Returns true iff the Jacobian matrix is not a function of x. **Note:** Jacobian matrix is either constant for both forward and inverse or neither.
`name`	Returns the string name of this `Bijector`.
`name_scope`	Returns a `tf.name_scope` (https://www.tensorflow.org/api_docs/python/tf/name_scope) instance for this class.
`non_trainable_variables`	Sequence of non-trainable variables owned by this module and its submodules. **Note:** this method uses reflection to find variables on the current instance and submodules. For performance reasons you may wish to cache the result of calling this method if you don't expect the return value to change.
`parameters`	Dictionary of parameters used to instantiate this `Bijector`.

`submodules`	Sequence of all sub-modules.
	Submodules are modules which are properties of this module, or found as properties of modules which are properties of this module (and so on).
	
	`>>> a = tf.Module()`
	`>>> b = tf.Module()`
	`>>> c = tf.Module()`
	`>>> a.b = b`
	`>>> b.c = c`
	`>>> list(a.submodules) == [b, c]`
	`True`
	`>>> list(b.submodules) == [c]`
	`True`
	`>>> list(c.submodules) == []`
	`True`

`trainable_variables`	Sequence of trainable variables owned by this module and its submodules. **Note:** this method uses reflection to find variables on the current instance and submodules. For performance reasons you may wish to cache the result of calling this method if you don't expect the return value to change.
`validate_args`	Returns True if Tensor arguments will be validated.
`variables`	Sequence of variables owned by this module and its submodules. **Note:** this method uses reflection to find variables on the current instance and submodules. For performance reasons you may wish to cache the result of calling this method if you don't expect the return value to change.

Methods

`forward`

View source (https://github.com/tensorflow/probability/blob/v0.13.0/tensorflow_probability/python/bijectors/invert.py#L116-L117)

```
forward(
    x, **kwargs
)
```

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

Args

`x`	`Tensor` (structure). The input to the 'forward' evaluation.
`name`	The name to give this op.
`**kwargs`	Named arguments forwarded to subclass implementation.

Returns

`Tensor` (structure).

Raises

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

forward_dtype

View source (https://github.com/tensorflow/probability/blob/v0.13.0/tensorflow_probability/python/bijectors/invert.py#L128-L129)

```
forward_dtype(
    dtype=bijector_lib.UNSPECIFIED, **kwargs
)
```

Returns the dtype returned by `forward` for the provided input.

forward_event_ndims

View source (https://github.com/tensorflow/probability/blob/v0.13.0/tensorflow_probability/python/bijectors/invert.py#L137-L138)

```
forward_event_ndims(
    event_ndims, **kwargs
)
```

Returns the number of event dimensions produced by `forward`.

forward_event_shape

View source (https://github.com/tensorflow/probability/blob/v0.13.0/tensorflow_probability/python/bijectors/invert.py#L89-L90)

```
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` (structure) indicating event-portion shape passed into `forward` function.

Returns

`forward_event_shape_tensor`	`TensorShape` (structure) indicating event-portion shape after applying `forward`. Possibly unknown.

forward_event_shape_tensor

View source (https://github.com/tensorflow/probability/blob/v0.13.0/tensorflow_probability/python/bijectors/invert.py#L92-L93)

```
forward_event_shape_tensor(
    input_shape
)
```

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

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

Returns	
`forward_event_shape_tensor`	`Tensor`, `int32` vector (structure) indicating event-portion shape after applying `forward`.

forward_log_det_jacobian

View source (https://github.com/tensorflow/probability/blob/v0.13.0/tensorflow_probability/python/bijectors/invert.py#L125-L126)

```
forward_log_det_jacobian(
    x, event_ndims=None, **kwargs
)
```

Returns both the forward_log_det_jacobian.

Args	
`x`	`Tensor` (structure). The input to the 'forward' Jacobian determinant evaluation.
`event_ndims`	Optional number of dimensions in the probabilistic events being transformed; this must be greater than or equal to `self.forward_min_event_ndims`. If `event_ndims` is specified, the log Jacobian determinant is summed to produce a scalar log-determinant for each event. Otherwise (if `event_ndims` is `None`), no

	reduction is performed. Multipart bijectors require *structured* event_ndims, such that the batch rank `rank(y[i]) - event_ndims[i]` is the same for all elements `i` of the structured input. In most cases (with the exception of `tfb.JointMap`) they further require that `event_ndims[i] - self.inverse_min_event_ndims[i]` is the same for all elements `i` of the structured input. Default value: `None` (equivalent to `self.forward_min_event_ndims`).
`name`	The name to give this op.
`**kwargs`	Named arguments forwarded to subclass implementation.

Returns

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

Raises

`TypeError`	if `y`'s dtype is incompatible with the expected output dtype.
`NotImplementedError`	if neither `_forward_log_det_jacobian` nor {`_inverse`, `_inverse_log_det_jacobian`} are implemented, or this is a non-injective bijector.
`ValueError`	if the value of `event_ndims` is not valid for this bijector.

inverse

View source (https://github.com/tensorflow/probability/blob/v0.13.0/tensorflow_probability/python/bijectors/invert.py#L119-L120)

```
inverse(
    y, **kwargs
)
```

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

Args

`y`	`Tensor` (structure). The input to the 'inverse' evaluation.

`name`	The name to give this op.
`**kwargs`	Named arguments forwarded to subclass implementation.

Returns

`Tensor` (structure), 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 `y`'s structured dtype is incompatible with the expected output dtype.
`NotImplementedError`	if `_inverse` is not implemented.

inverse_dtype

View source (https://github.com/tensorflow/probability/blob/v0.13.0/tensorflow_probability/python/bijectors/invert.py#L131-L132)

```
inverse_dtype(
    dtype=bijector_lib.UNSPECIFIED, **kwargs
)
```

Returns the dtype returned by `inverse` for the provided input.

inverse_event_ndims

View source (https://github.com/tensorflow/probability/blob/v0.13.0/tensorflow_probability/python/bijectors/invert.py#L134-L135)

```
inverse_event_ndims(
    event_ndims, **kwargs
)
```

Returns the number of event dimensions produced by `inverse`.

inverse_event_shape

View source (https://github.com/tensorflow/probability/blob/v0.13.0/tensorflow_probability/python/bijectors/invert.py#L95-L96)

```
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` (structure) indicating event-portion shape passed into `inverse` function.

Returns

`inverse_event_shape_tensor`	`TensorShape` (structure) indicating event-portion shape after applying `inverse`. Possibly unknown.

inverse_event_shape_tensor

View source (https://github.com/tensorflow/probability/blob/v0.13.0/tensorflow_probability/python/bijectors/invert.py#L98-L99)

```
inverse_event_shape_tensor(
    output_shape
)
```

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

Args

`output_shape`	`Tensor`, `int32` vector (structure) indicating event-portion shape passed into `inverse` function.

`name`	name to give to the op

Returns

`inverse_event_shape_tensor`	`Tensor`, `int32` vector (structure) indicating event-portion shape after applying `inverse`.

inverse_log_det_jacobian

View source (https://github.com/tensorflow/probability/blob/v0.13.0/tensorflow_probability/python/bijectors/invert.py#L122-L123)

```
inverse_log_det_jacobian(
    y, event_ndims=None, **kwargs
)
```

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` (structure). The input to the 'inverse' Jacobian determinant evaluation.
`event_ndims`	Optional number of dimensions in the probabilistic events being transformed; this must be greater than or equal to `self.inverse_min_event_ndims`. If `event_ndims` is specified, the log Jacobian determinant is summed to produce a scalar log-determinant for each event. Otherwise (if `event_ndims` is `None`), no reduction is performed. Multipart bijectors require *structured* event_ndims, such that the batch rank `rank(y[i]) - event_ndims[i]` is the same for all elements `i` of the structured input. In most cases (with the exception of `tfb.JointMap`) they further require that `event_ndims[i] - self.inverse_min_event_ndims[i]` is the same for all elements `i` of the structured input. Default value: `None` (equivalent to `self.inverse_min_event_ndims`).

`name`	The name to give this op.
`**kwargs`	Named arguments forwarded to subclass implementation.

Returns

`ildj`	`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 `x`'s dtype is incompatible with the expected inverse-dtype.
`NotImplementedError`	if `_inverse_log_det_jacobian` is not implemented.
`ValueError`	if the value of `event_ndims` is not valid for this bijector.

parameter_properties

View source (https://github.com/tensorflow/probability/blob/v0.13.0/tensorflow_probability/python/bijectors/bijector.py#L1077-L1096)

```
@classmethod
parameter_properties(
    dtype=tf.float32
)
```

Returns a dict mapping constructor arg names to property annotations.

This dict should include an entry for each of the bijector's `Tensor`-valued constructor arguments.

Args

`dtype`	Optional float `dtype` to assume for continuous-valued parameters. Some constraining bijectors require advance knowledge of the dtype because certain constants (e.g., `tfb.Softplus.low`) must be instantiated with the same dtype as the values to be transformed.

Returns

`parameter_properties`	A `str -` `>`tfp.python.internal.parameter_properties.ParameterProperties`dict mapping constructor argument names to`ParameterProperties` instances.

with_name_scope

```
@classmethod
with_name_scope(
    method
)
```

Decorator to automatically enter the module name scope.

```
>>> class MyModule(tf.Module):
...   @tf.Module.with_name_scope
...   def __call__(self, x):
...     if not hasattr(self, 'w'):
...       self.w = tf.Variable(tf.random.normal([x.shape[1], 3]))
...     return tf.matmul(x, self.w)
```

Using the above module would produce `tf.Variable` (https://www.tensorflow.org/api_docs/python/tf/Variable)s and `tf.Tensor` (https://www.tensorflow.org/api_docs/python/tf/Tensor)s whose names included the module name:

```
>>> mod = MyModule()
>>> mod(tf.ones([1, 2]))
<tf.Tensor: shape=(1, 3), dtype=float32, numpy=..., dtype=float32)>
>>> mod.w
<tf.Variable 'my_module/Variable:0' shape=(2, 3) dtype=float32,
numpy=..., dtype=float32)>
```

Args

`method`	The method to wrap.

Returns

The original method wrapped such that it enters the module's name scope.

`__call__`

View source (https://github.com/tensorflow/probability/blob/v0.13.0/tensorflow_probability/python/bijectors/bijector.py#L859-L944)

```
__call__(
    value, name=None, **kwargs
)
```

Applies or composes the `Bijector`, depending on input type.

This is a convenience function which applies the `Bijector` instance in three different ways, depending on the input:

1. If the input is a `tfd.Distribution` instance, return `tfd.TransformedDistribution(distribution=input, bijector=self)`.
2. If the input is a `tfb.Bijector` instance, return `tfb.Chain([self, input])`.
3. Otherwise, return `self.forward(input)`

Args

`value`	A `tfd.Distribution`, `tfb.Bijector`, or a (structure of) `Tensor`.
`name`	Python `str` name given to ops created by this function.
`**kwargs`	Additional keyword arguments passed into the created `tfd.TransformedDistribution`, `tfb.Bijector`, or `self.forward`.

Returns

`composition`	A `tfd.TransformedDistribution` if the input was a `tfd.Distribution`, a `tfb.Chain` if the input was a `tfb.Bijector`, or a (structure of) `Tensor` computed by `self.forward`.

Examples

```
sigmoid = tfb.Reciprocal()(
    tfb.Shift(shift=1.)(
      tfb.Exp()(
        tfb.Scale(scale=-1.))))
# ==> `tfb.Chain([
#         tfb.Reciprocal(),
#         tfb.Shift(shift=1.),
#         tfb.Exp(),
#         tfb.Scale(scale=-1.),
#      ])`  # ie, `tfb.Sigmoid()`

log_normal = tfb.Exp()(tfd.Normal(0, 1))
# ==> `tfd.TransformedDistribution(tfd.Normal(0, 1), tfb.Exp())`

tfb.Exp()([-1., 0., 1.])
# ==> tf.exp([-1., 0., 1.])
```

__eq__

View source (https://github.com/tensorflow/probability/blob/v0.13.0/tensorflow_probability/python/bijectors/bijector.py#L818-L849)

```
__eq__(
    other
)
```

Return self==value.