InferPy Documentation

Release 1.0

Javier Cózar, Rafael Cabañas, Antonio Salmerón, Andrés R. Mase

Jul 10, 2019

QUICK START

InferPy is a high-level API for probabilistic modeling written in Python and capable of running on top of Tensorflow. InferPy's API is strongly inspired by Keras and it has a focus on enabling flexible data processing, easy-to-code probablistic modeling, scalable inference and robust model validation.

Use InferPy if you need a probabilistic programming language that:

- Allows easy and fast prototyping of hierarchical probabilistic models with a simple and user friendly API inspired by Keras.
- Automatically creates computational efficient batched models without the need to deal with complex tensor operations.
- Run seamlessly on CPU and GPU by relying on Tensorflow, without having to learn how to use Tensorflow.

A set of examples can be found in the Probabilistic Model Zoo section.

CHAPTER

ONE

GETTING STARTED:

1.1 Installation

Install InferPy from PyPI:

```
$ python -m pip install inferpy
```

1.2 30 seconds to InferPy

The core data structures of InferPy is a **probabilistic model**, defined as a set of **random variables** with a conditional dependency structure. A **random varible** is an object parameterized by a set of tensors.

Let's look at a simple non-linear **probabilistic component analysis** model (NLPCA). Graphically the model can be defined as follows,

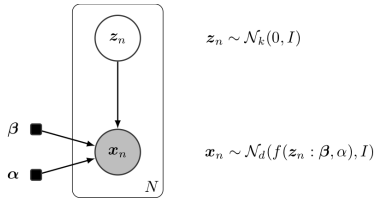


Fig. 1: Non-linear PCA

We start by importing the required packages and defining the constant parameters in the model.

```
import inferpy as inf
import tensorflow as tf

# number of components
k = 1
# size of the hidden layer in the NN
d0 = 100
# dimensionality of the data
dx = 2
# number of observations (dataset size)
N = 1000
```

A model can be defined by decorating any function with `@inf.probmodel`. The model is fully specified by the variables defined inside this function:

```
@inf.probmodel
def nlpca(k, d0, dx, decoder):

    with inf.datamodel():
        z = inf.Normal(tf.ones([k])*0.5, 1., name="z")     # shape = [N,k]
        output = decoder(z,d0,dx)
        x_loc = output[:,:dx]
        x_scale = tf.nn.softmax(output[:,dx:])
        x = inf.Normal(x_loc, x_scale, name="x")    # shape = [N,d]
```

The construct `with inf.datamodel()`, which resembles to the **plateau notation**, will replicate N times the variables enclosed, where N is the size of our data.

In the previous model, the input argument `decoder` must be a function implementing a neural network. This might be defined outside the model as follows.

```
def decoder(z,d0,dx):
    h0 = tf.layers.dense(z, d0, tf.nn.relu)
    return tf.layers.dense(h0, 2 * dx)
```

Now, we can instantiate our model and obtain samples (from the prior distributions).

```
# create an instance of the model
m = nlpca(k,d0,dx, decoder)

# Sample from priors
samples = m.sample()
```

In variational inference, we must defined a Q-model as follows.

```
@inf.probmodel
def qmodel(k):
    with inf.datamodel():
        qz_loc = inf.Parameter(tf.ones([k])*0.5, name="qz_loc")
        qz_scale = tf.math.softplus(inf.Parameter(tf.ones([k]),name="qz_scale"))

        qz = inf.Normal(qz_loc, qz_scale, name="z")
```

Afterwards, we define the parameters of our inference algorithm and fit the data to the model.

```
# set the inference algorithm
VI = inf.inference.VI(qmodel(k), epochs=5000)

# learn the parameters
m.fit({"x": x_train}, VI)
```

The inference method can be further configure. But, as in Keras, a core principle is to try make things reasonably simple, while allowing the user the full control if needed.

Finally, we might extract the posterior of `z`, which is basically the hidden representation of our data.

```
#extract the hidden representation
hidden_encoding = m.posterior["z"]
print(hidden_encoding.sample())
```

CHAPTER

TWO

GUIDING PRINCIPLES

2.1 Features

The main features of InferPy are listed below.

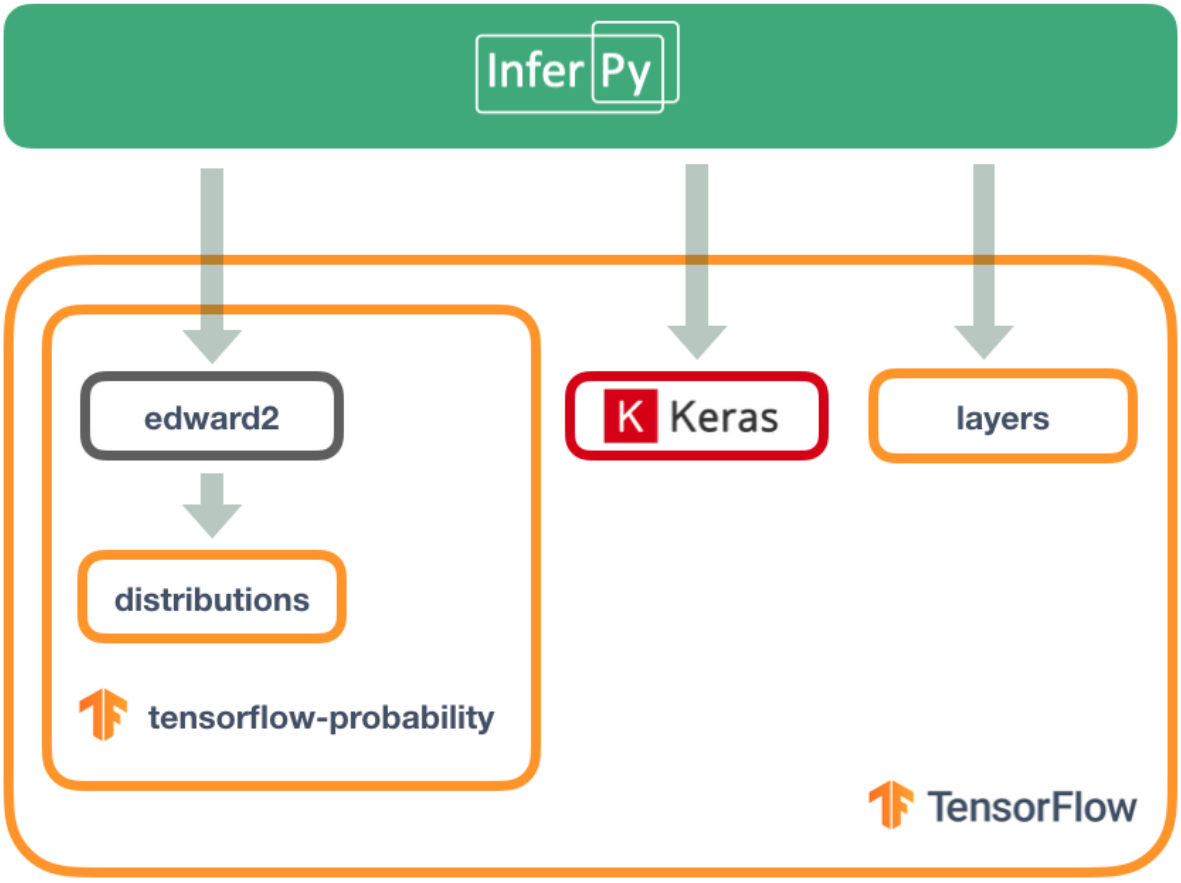
- Allows a simple definition and inference of probabilistic models containing deep neural networks.
- The models that can be defined in InferPy are those that can be defined using Edward2 (i.e., `tfp.edward2`, whose probability distribuions are mainly inherited from the module `distributions` in the tensorflow-probability package.
- Edward's drawback is that for the model definition, the user has to manage complex multidimensional arrays called tensors. By contrast, in InferPy all the parameters in a model can be defined using the standard Python types (compatibility with Numpy is available as well).
- InferPy directly relies on top of Edward's inference engine and includes all the inference algorithms included in this package. As Edward's inference engine relies on TensorFlow computing engine, InferPy also relies on it too.
- Unlike Edward, our package does not require to have a strong background in the inference methods.

2.2 Architecture

Given the previous considerations, we might summarize the InferPy architecture as follows.

Note that InferPy can be seen as an upper layer for working with probabilistic distributions defined over tensors. Most of the interaction is done with Edward: the definitions of the random variables, the inference. However, InferPy also interacts directly with TensorFlow in some operations that are hidden to the user, e.g., the manipulation of the tensors representing the parameters of the distributions.

An additional advantage of using Edward and TensorFlow as inference engine, is that all the paralelisation details are hidden to the user. Moreover, the same code will run either in CPUs or GPUs.


InferPy
edward2
distributions
tensorflow-probability
Keras
layers
TensorFlow

CHAPTER

THREE

REQUIREMENTS

3.1 System

Currently, InferPy requires Python 3.5 or higher. For checking your default Python version, type:

```
$ python --version
```

Travis tests are performed on versions 3.5 and 3.6. Go to https://www.python.org/ for specific instructions for installing the Python interpreter in your system.

InferPy runs in any OS with the Python interpreter installed. In particular, tests have been carried out for the systems listed bellow.

- Linux CentOS 7
- Linux Elementary 0.4
- Linux Mint 19
- Linux Ubuntu 14.04 16.04 18.04
- MacOS High Sierra (10.13) and Mojave (10.14)
- Windows 10 Enterprise

3.2 Package Dependencies

For a basic usage, InferPy requires the following packages:

```
tensorflow>=1.12.1,<2.0
tensorflow-probability>=0.5.0,<1.0
networkx>=2.2.0<3.0
matplotlib>=2.2.3,<3.0
Keras==2.2.4
Keras-Applications==1.0.7
Keras-Preprocessing==1.0.9
protobuf==3.8.0
```

CHAPTER

FOUR

GUIDE TO PROBABILISTIC MODELS

4.1 Getting Started with Probabilistic Models

InferPy focuses on *hierarchical probabilistic models* structured in two different layers:

- A **prior model** defining a joint distribution $p(\mathbf{w})$ over the global parameters of the model. $\mathbf{w}$ can be a single random variable or a bunch of random variables with any given dependency structure.
- A **data or observation model** defining a joint conditional distribution $p(\mathbf{x}, \mathbf{z}|\mathbf{w})$ over the observed quantities $\mathbf{x}$ and the the local hidden variables $\mathbf{z}$ governing the observation $\mathbf{x}$. This data model is specified in a single-sample basis. There are many models of interest without local hidden variables, in that case, we simply specify the conditional $p(\mathbf{x}|\mathbf{w})$. Similarly, either $\mathbf{x}$ or $\mathbf{z}$ can be a single random variable or a bunch of random variables with any given dependency structure.

For example, a Bayesian PCA model has the following graphical structure,

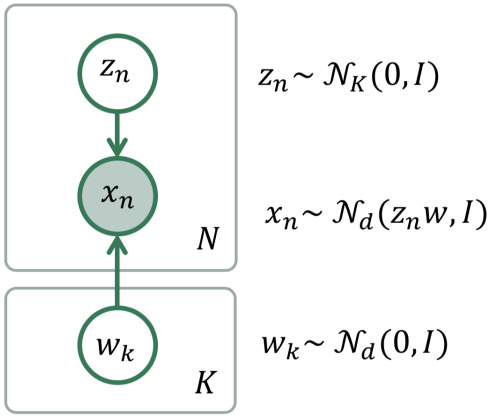


Fig. 1: Bayesian PCA

The **prior model** are the variables w_k. The **data model** is the part of the model surrounded by the box indexed by **N**.

And this is how this Bayesian PCA model is denfined in InferPy:

```
# definition of a generic model
@inf.probmodel
def pca(k,d):
    w = inf.Normal(loc=np.zeros([k,d]), scale=1, name="w")       # shape = [k,d]
    with inf.datamodel():
        z = inf.Normal(np.ones(k),1, name="z")                 # shape = [N,k]
        x = inf.Normal(z @ w , 1, name="x")                      # shape = [N,d]
```

(continues on next page)

(continued from previous page)

```
# create an instance of the model
m = pca(k=1,d=2)
```

The `with inf.datamodel()` sintaxis is used to replicate the random variables contained within this construct. It follows from the so-called *plateau notation* to define the data generation part of a probabilistic model. Every replicated variable is **conditionally idependent** given the previous random variables (if any) defined outside the **with** statement. The plateau size will be later automatically calculated, so there is not need to specify it. Yet, this construct has an optional input parameter for specifying its size, e.g., `with inf.datamodel(size=N)`. This should be consistent with the size of our data.

4.2 Random Variables

Any random variable in InferPy encapsulates an equivalent one in Edward 2, and hence it also has associated a distribution object from TensorFlow Probability. These can be accessed using the properties `var` and `distribution` respectively:

```
>>> x = inf.Normal(loc = 0, scale = 1)

>>> x.var
<ed.RandomVariable 'randvar_0/' shape=() dtype=float32>

>>> x.distribution
<tfp.distributions.Normal 'randvar_0/' batch_shape=() event_shape=() dtype=float32>
```

Even more, InferPy random variables inherit all the properties and methods from Edward2 variables or TensorFlow Probability distributions (in this order or priority). For example:

```
>>> x.value
<tf.Tensor 'randvar_0/sample/Reshape:0' shape=() dtype=float32>

>>> x.sample()
-0.05060442

>>> x.loc
<tf.Tensor 'randvar_0/Identity:0' shape=() dtype=float32>
```

In the previous code, `value` is inherited form the encapsulated Edward2 object while `sample()` and the parameter `loc` are obtained from the distribution object. Note that the method `sample()` returns an evaluated tensors. In case of desiring it not to be evaluated, simply use the input parameter `tf_run` as follows.

```
>>> x.sample(tf_run=False)
<tf.Tensor 'randvar_0/sample/Reshape:0' shape=() dtype=float32>
```

Following Edward's approach, we (conceptually) partition a random variable's shape into three groups:

- *Batch shape* describes independent, not identically distributed draws. Namely, we may have a set of (different) parameterizations to the same distribution.
- *Sample shape* describes independent, identically distributed draws from the distribution.
- *Event shape* describes the shape of a single draw (event space) from the distribution; it may be dependent across dimensions.

The previous attributes can be accessed by `x.batch_shape`, `x.sample_shape` and `x.event_shape`, respectively. When declaring random variables, the *batch_shape* is obtained from the distribution parameters. For as long as possible, the parameters will be broadcasted. With this in mind, all the definitions in the following code are equivalent.

```
x = inf.Normal(loc = [[0.,0.],[0.,0.],[0.,0.]], scale=1)  # x.shape = [3,2]

x = inf.Normal(loc = np.zeros([3,2]), scale=1)            # x.shape = [3,2]

x = inf.Normal(loc = 0, scale=tf.ones([3,2]))             # x.shape = [3,2]
```

The `sample_shape` can be explicitly stated using the input parameter sample_shape, but this only can be done outside a model definition. Inside of `inf.probmodels`, the sample_shape is fixed by `with inf.datamodel(size = N)` (using the size argument when provided, or in runtime depending on the observed data).

```
x = inf.Normal(tf.ones([3,2]), 0, sample_shape=100)      # x.sample = [100,3,2]

with inf.datamodel(100):
    x = inf.Normal(tf.ones([3, 2]), 0)                   # x.sample = [100,3,2]
```

Finally, the *event shape* will only be consider in some distributions. This is the case of the multivariate Gaussian:

```
x = inf.MultivariateNormalDiag(loc=[1., -1], scale_diag=[1, 2.])
```

```
>>> x.event_shape
TensorShape([Dimension(2)])

>>> x.batch_shape
TensorShape([])

>>> x.sample_shape
TensorShape([])
```

Note that indexing over all the defined dimenensions is supported:

```
with inf.datamodel(size=10):
    x = inf.models.Normal(loc=tf.zeros(5), scale=1.)       # x.shape = [10,5]


y = x[7,4]                                                 # y.shape = []

y2 = x[7]                                                  # y2.shape = [5]

y3 = x[7,:]                                                # y2.shape = [5]

y4 = x[:,4]                                                # y4.shape = [10]
```

Moreover, we may use indexation for defining new variables whose indexes may be other (discrete) variables.

```
i = inf.Categorical(logits= tf.zeros(3))          # shape = []
mu = inf.Normal([5,1,-2], 0.)                     # shape = [3]
x = inf.Normal(mu[i], scale=1.)                   # shape = []
```

4.3 Probabilistic Models

A **probabilistic model** defines a joint distribution over observable and hidden variables, i.e., $p(\mathbf{w}, \mathbf{z}, \mathbf{x})$. Note that a variable might be observable or hidden depending on the fitted data. Thus this is not specified when defining the model.

A probabilistic model is defined by decorating any function with `@inf.probmodel`. The model is made of any variable defined inside this function. A simple example is shown below.

```
@inf.probmodel
def simple(mu=0):
    # global variables
    theta = inf.Normal(mu, 0.1, name="theta")

    # local variables
    with inf.datamodel():
        x = inf.Normal(theta, 1, name="x")
```

Note that any variable in a model can be initialized with a name. If not provided, names generated automatically will be used. However, it is highly convenient to explicitly specify the name of a random variable because in this way it will be able to be referenced in some inference stages.

The model must be **instantiated** before it can be used. This is done by simple invoking the function (which will return a probmodel object).

```
>>> m = simple()
>>> type(m)
<class 'inferpy.models.prob_model.ProbModel'>
```

Now we can use the model with the prior probabilities. For example, we might get a sample or access to the distribution parameters:

```
>>> m.prior().sample()
{'theta': -0.074800275, 'x': array([0.07758344], dtype=float32)}

>>> m.prior().parameters()
{'theta': {'name': 'theta',
  'allow_nan_stats': True,
  'validate_args': False,
  'scale': 0.1,
  'loc': 0},
 'x': {'name': 'x',
  'allow_nan_stats': True,
  'validate_args': False,
  'scale': 1,
  'loc': 0.116854645}}
```

or to extract the variables:

```
>>> m.vars["theta"]
<inf.RandomVariable (Normal distribution) named theta/, shape=(), dtype=float32>
```

We can create new and different instances of our model:

```
>>> m2 = simple(mu=5)
>>> m==m2
False
```

4.4 Supported Probability Distributions

Supported probability distributions are located in the package `inferpy.models`. All of them have `inferpy.models.RandomVariable` as superclass. A list with all the supported distributions can be obtained as as follows.

```
>>> inf.models.random_variable.distributions_all
['Autoregressive', 'BatchReshape', 'Bernoulli', 'Beta', 'BetaWithSoftplusConcentration
↪',
 'Binomial', 'Categorical', 'Cauchy', 'Chi2', 'Chi2WithAbsDf',
↪'ConditionalTransformedDistribution',
  'Deterministic', 'Dirichlet', 'DirichletMultinomial', 'ExpRelaxedOneHotCategorical',
↪ '
  Exponential', 'ExponentialWithSoftplusRate', 'Gamma', 'GammaGamma',
  'GammaWithSoftplusConcentrationRate', 'Geometric', 'GaussianProcess',
  'GaussianProcessRegressionModel', 'Gumbel', 'HalfCauchy', 'HalfNormal',
  'HiddenMarkovModel', 'Horseshoe', 'Independent', 'InverseGamma',
   'InverseGammaWithSoftplusConcentrationRate', 'InverseGaussian', 'Kumaraswamy',
   'LinearGaussianStateSpaceModel', 'Laplace', 'LaplaceWithSoftplusScale', 'LKJ',
  'Logistic', 'LogNormal', 'Mixture', 'MixtureSameFamily', 'Multinomial',
   'MultivariateNormalDiag', 'MultivariateNormalFullCovariance',
↪'MultivariateNormalLinearOperator',
   'MultivariateNormalTriL', 'MultivariateNormalDiagPlusLowRank',
↪'MultivariateNormalDiagWithSoftplusScale',
   'MultivariateStudentTLinearOperator', 'NegativeBinomial', 'Normal',
↪'NormalWithSoftplusScale',
   'OneHotCategorical', 'Pareto', 'Poisson', 'PoissonLogNormalQuadratureCompound',
↪'QuantizedDistribution',
   'RelaxedBernoulli', 'RelaxedOneHotCategorical', 'SinhArcsinh', 'StudentT',
↪'StudentTWithAbsDfSoftplusScale',
   'StudentTProcess', 'TransformedDistribution', 'Triangular', 'TruncatedNormal',
↪'Uniform', 'VectorDeterministic',
   'VectorDiffeomixture', 'VectorExponentialDiag', 'VectorLaplaceDiag',
↪'VectorSinhArcsinhDiag', 'VonMises',
   'VonMisesFisher', 'Wishart', 'Zipf']
```

Note that these are all the distributions in Edward 2 and hence in TensorFlow Probability. Their input parameters will be the same.

CHAPTER

FIVE

GUIDE TO APPROXIMATE INFERENCE

5.1 Variational Inference

The API defines the set of algorithms and methods used to perform inference in a probabilistic model $p(x, z, \theta)$ (where x are the observations, z the local hidden variables, and θ the global parameters of the model). More precisely, the inference problem reduces to compute the posterior probability over the latent variables given a data sample $p(z, \theta|x_{train})$, because by looking at these posteriors we can uncover the hidden structure in the data. Let us consider the following model:

```
@inf.probmodel
def pca(k,d):
    w = inf.Normal(loc=tf.zeros([k,d]), scale=1, name="w")       # shape = [k,d]
    with inf.datamodel():
        z = inf.Normal(tf.ones([k]),1, name="z")                 # shape = [N,k]
        x = inf.Normal(z @ w , 1, name="x")                      # shape = [N,d]
```

In this model, the posterior over the local hidden variables $p(w_n|x_{train})$ tell us the latent vector representation of the sample x_n, while the posterior over the global variables $p(\mu|x_{train})$ tells us which is the affine transformation between the latent space and the observable space.

InferPy inherits Edward's approach an consider approximate inference solutions,

$$q(z, \theta) \approx p(z, \theta|x_{train})$$

in which the task is to approximate the posterior $p(z, \theta|x_{train})$ using a family of distributions, $q(z, \theta; \lambda)$, indexed by a parameter vector λ.

For making inference, we must define a model 'Q' for approximating the posterior distribution. This is also done by defining a function decorated with `@inf.probmodel`:

```
@inf.probmodel
def qmodel(k,d):
    qw_loc = inf.Parameter(tf.ones([k,d]), name="qw_loc")
    qw_scale = tf.math.softplus(inf.Parameter(tf.ones([k, d]), name="qw_scale"))
    qw = inf.Normal(qw_loc, qw_scale, name="w")

    with inf.datamodel():
        qz_loc = inf.Parameter(tf.ones([k]), name="qz_loc")
        qz_scale = tf.math.softplus(inf.Parameter(tf.ones([k]), name="qz_scale"))
        qz = inf.Normal(qz_loc, qz_scale, name="z")
```

In the 'Q' model we should include a q distribution for every non observed variable in the 'P' model. These varaiables are also objects of class `inferpy.RandomVariable`. However, their parameters might be of type `inf.Parameter`, which are objects encapsulating TensorFlow trainable variables.

Then, we set the parameters of the inference algorithm. In case of variational inference (VI) we must specify an instance of the 'Q' model and the number of `epochs` (i.e., iterations). For example:

```
# set the inference algorithm
VI = inf.inference.VI(qmodel(k=1,d=2), epochs=1000)
```

VI can be further configured by setting the parameter `optimizer` which indicates the TensorFlow optimizer to be used (AdamOptimizer by default).

Stochastic VI is similarly specified but has an additional input parameter for specifying the batch size:

```
SVI = inf.inference.SVI(qmodel(k=1,d=2), epochs=1000, batch_size=200)
```

Then we must instantiate our 'P' model and fit the data with the inference algorithm defined.

```
# create an instance of the model
m = pca(k=1,d=2)
# run the inference
m.fit({"x": x_train}, VI)
```

The output generated will be similar to:

```
 0 epochs         44601.14453125.....................
 200 epochs         44196.98046875.....................
 400 epochs         50616.359375.....................
 600 epochs         41085.6484375.....................
 800 epochs         30349.79296875.....................
```

Finally we can access to the parameters of the posterior distributions:

```
>>> m.posterior("w").parameters()
{'name': 'w',
 'allow_nan_stats': True,
 'validate_args': False,
 'scale': array([[0.9834974 , 0.99731755]], dtype=float32),
 'loc': array([[1.7543027, 1.7246702]], dtype=float32)}
```

5.2 Custom Loss function

Following InferPy guiding principles, users can further configure the inference algorithm. For example, we might be interested in defining our own function to minimise. As an example, we define the following function taking as input parameters the random variables of the P and Q models (we assume that their sample sizes are consistent with the plates in the mdoel). Note that the output of this function must be a tensor.

```
def custom_elbo(pvars, qvars, **kwargs):

    # compute energy
    energy = tf.reduce_sum([tf.reduce_sum(p.log_prob(p.value)) for p in pvars.
↪values()])

    # compute entropy
    entropy = - tf.reduce_sum([tf.reduce_sum(q.log_prob(q.value)) for q in qvars.
↪values()])

    # compute ELBO
```

(continues on next page)

(continued from previous page)

```
    ELBO = energy + entropy

    # This function will be minimized. Return minus ELBO
    return -ELBO
```

For using our own loss function, we simply have to pass this function to the input parameter `loss` in the inference method constructor. For example:

```
# set the inference algorithm
VI = inf.inference.VI(qmodel(k=1,d=2), loss=custom_elbo, epochs=1000)

# run the inference
m.fit({"x": x_train}, VI)
```

After this, the rest of the code remains unchanged.

CHAPTER

SIX

GUIDE TO BAYESIAN DEEP LEARNING

InferPy inherits Edward's approach for representing probabilistic models as (stochastic) computational graphs. As describe above, a random variable x is associated to a tensor x^* in the computational graph handled by TensorFlow, where the computations takes place. This tensor x^* contains the samples of the random variable x, i.e. $x^* \sim p(x|\theta)$. In this way, random variables can be involved in complex deterministic operations containing deep neural networks, math operations and another libraries compatible with Tensorflow (such as Keras).

Bayesian deep learning or deep probabilistic programming enbraces the idea of employing deep neural networks within a probabilistic model in order to capture complex non-linear dependencies between variables.

InferPy's API gives support to this powerful and flexible modeling framework. Let us start by showing how a non-linear PCA can be defined by mixing `tf.layers` and InferPy code.

```
import inferpy as inf
import tensorflow as tf


# number of components
k = 1
# size of the hidden layer in the NN
d0 = 100
# dimensionality of the data
dx = 2
# number of observations (dataset size)
N = 1000


@inf.probmodel
def nlpca(k, d0, dx, decoder):

    with inf.datamodel():
        z = inf.Normal(tf.ones([k])*0.5, 1., name="z")     # shape = [N,k]
        output = decoder(z,d0,dx)
        x_loc = output[:,:dx]
        x_scale = tf.nn.softmax(output[:,dx:])
        x = inf.Normal(x_loc, x_scale, name="x")    # shape = [N,d]


def decoder(z,d0,dx):
    h0 = tf.layers.dense(z, d0, tf.nn.relu)
    return tf.layers.dense(h0, 2 * dx)


# Q-model  approximating P
```

(continues on next page)

(continued from previous page)

```
@inf.probmodel
def qmodel(k):
    with inf.datamodel():
        qz_loc = inf.Parameter(tf.ones([k])*0.5, name="qz_loc")
        qz_scale = tf.math.softplus(inf.Parameter(tf.ones([k]),name="qz_scale"))

        qz = inf.Normal(qz_loc, qz_scale, name="z")


# create an instance of the model
m = nlpca(k,d0,dx, decoder)

# set the inference algorithm
VI = inf.inference.VI(qmodel(k), epochs=5000)

# learn the parameters
m.fit({"x": x_train}, VI)

#extract the hidden representation
hidden_encoding = m.posterior("z")
print(hidden_encoding.sample())
```

In this case, the parameters of the decoder neural network (i.e., weights) are automatically managed by TensorFlow. These parameters are them treated as model parameters and not exposed to the user. In consequence, we can not be Bayesian about them by defining specific prior distributions.

Alternatively, we could use Keras layers by simply defining an alternative decoder function as follows.

```
def decoder_keras(z,d0,dx):
    h0 = tf.keras.layers.Dense(d0, activation=tf.nn.relu, name="encoder_h0")
    h1 = tf.keras.layers.Dense(2*dx, name="encoder_h1")
    return h1(h0(z))


# create an instance of the model
m = nlpca(k,d0,dx, decoder_keras)
```

CHAPTER

SEVEN

PROBABILISTIC MODEL ZOO

In this section, we present the code for implementing some models in Inferpy.

7.1 Bayesian Linear Regression

Graphically, a (Bayesian) linear regression can be defined as follows,

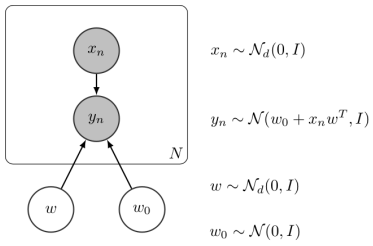


Fig. 1: Bayesian Linear Regression

The InferPy code for this model is shown below,

```
import inferpy as inf
import tensorflow as tf


@inf.probmodel
def linear_reg(d):
    w0 = inf.Normal(0, 1, name="w0")
    w = inf.Normal(tf.zeros([d, 1]), 1, name="w")

    with inf.datamodel():
        x = inf.Normal(tf.ones(d), 2, name="x")
        y = inf.Normal(w0 + x @ w, 1.0, name="y")


@inf.probmodel
def qmodel(d):
    qw0_loc = inf.Parameter(0., name="qw0_loc")
    qw0_scale = tf.math.softplus(inf.Parameter(1., name="qw0_scale"))
    qw0 = inf.Normal(qw0_loc, qw0_scale, name="w0")

    qw_loc = inf.Parameter(tf.zeros([d, 1]), name="qw_loc")
    qw_scale = tf.math.softplus(inf.Parameter(tf.ones([d, 1]), name="qw_scale"))
```

(continues on next page)

(continued from previous page)

```
    qw = inf.Normal(qw_loc, qw_scale, name="w")


# create an instance of the model
m = linear_reg(d=2)

# create toy data
N = 1000
data = m.prior(["x", "y"], data={"w0": 0, "w": [[2], [1]]}).sample(N)

x_train = data["x"]
y_train = data["y"]

# set and run the inference
VI = inf.inference.VI(qmodel(2), epochs=10000)
m.fit({"x": x_train, "y": y_train}, VI)


# extract the parameters of the posterior
m.posterior(["w", "w0"]).parameters()
```

7.2 Bayesian Logistic Regression

Graphically, a (Bayesian) logistic regression can be defined as follows,

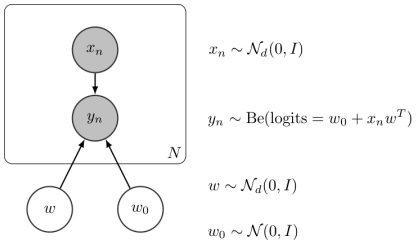


Fig. 2: Bayesian Linear Regression

The InferPy code for this model is shown below,

```
# required pacakges
import inferpy as inf
import numpy as np
import tensorflow as tf


@inf.probmodel
def log_reg(d):
    w0 = inf.Normal(0., 1., name="w0")
    w = inf.Normal(np.zeros([d, 1]), np.ones([d, 1]), name="w")

    with inf.datamodel():
        x = inf.Normal(np.zeros(d), 2., name="x")  # the scale is broadcasted to␣
↪shape [d] because of loc
```

(continues on next page)

(continued from previous page)

```
        y = inf.Bernoulli(logits=w0 + x @ w, name="y")


@inf.probmodel
def qmodel(d):
    qw0_loc = inf.Parameter(0., name="qw0_loc")
    qw0_scale = tf.math.softplus(inf.Parameter(1., name="qw0_scale"))
    qw0 = inf.Normal(qw0_loc, qw0_scale, name="w0")

    qw_loc = inf.Parameter(tf.zeros([d, 1]), name="qw_loc")
    qw_scale = tf.math.softplus(inf.Parameter(tf.ones([d, 1]), name="qw_scale"))
    qw = inf.Normal(qw_loc, qw_scale, name="w")


# create an instance of the model
m = log_reg(d=2)

# create toy data
N = 1000
data = m.prior(["x", "y"], data={"w0": 0, "w": [[2], [1]]}).sample(N)
x_train = data["x"]
y_train = data["y"]

VI = inf.inference.VI(qmodel(2), epochs=10000)
m.fit({"x": x_train, "y": y_train}, VI)

sess = inf.get_session()
print(m.posterior("w").sample())
print(m.posterior("w").parameters())
```

7.3 Linear Factor Model (PCA)

A linear factor model allows to perform principal component analysis (PCA). Graphically, it can be defined as follows,

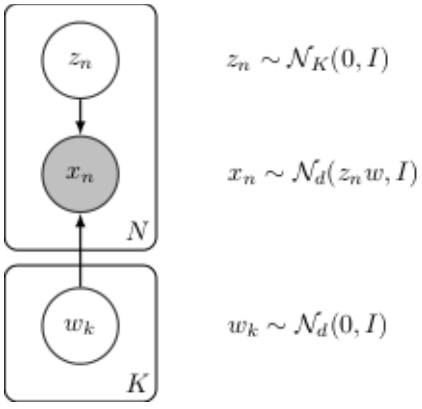


Fig. 3: Linear Factor Model (PCA)

The InferPy code for this model is shown below,

```
# Generate toy data
x_train = np.concatenate([
    inf.Normal([0.0, 0.0], scale=1.).sample(int(N/2)),
    inf.Normal([10.0, 10.0], scale=1.).sample(int(N/2))
    ])
x_test = np.concatenate([
    inf.Normal([0.0, 0.0], scale=1.).sample(int(N/2)),
    inf.Normal([10.0, 10.0], scale=1.).sample(int(N/2))
    ])


# definition of a generic model
@inf.probmodel
def pca(k, d):
    beta = inf.Normal(loc=tf.zeros([k, d]),
                      scale=1, name="beta")                # shape = [k,d]

    with inf.datamodel():
        z = inf.Normal(tf.ones(k), 1, name="z")        # shape = [N,k]
        x = inf.Normal(z @ beta, 1, name="x")          # shape = [N,d]


@inf.probmodel
def qmodel(k, d):
    qbeta_loc = inf.Parameter(tf.zeros([k, d]), name="qbeta_loc")
    qbeta_scale = tf.math.softplus(inf.Parameter(tf.ones([k, d]),
                                                 name="qbeta_scale"))

    qbeta = inf.Normal(qbeta_loc, qbeta_scale, name="beta")

    with inf.datamodel():
        qz_loc = inf.Parameter(np.ones(k), name="qz_loc")
        qz_scale = tf.math.softplus(inf.Parameter(tf.ones(k),
                                                  name="qz_scale"))

        qz = inf.Normal(qz_loc, qz_scale, name="z")


# create an instance of the model and qmodel
m = pca(k=1, d=2)
q = qmodel(k=1, d=2)

# set the inference algorithm
VI = inf.inference.VI(q, epochs=2000)

# learn the parameters
m.fit({"x": x_train}, VI)

# extract the hidden encoding
```

7.4 Non-linear Factor Model (NLPCA)

Similarly to the previous model, the Non-linear PCA can be graphically defined as follows,

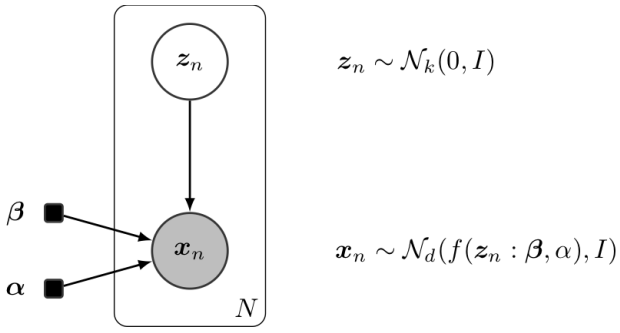


Fig. 4: Non-linear PCA

Its code in InferPy is shown below,

```
import inferpy as inf
import tensorflow as tf

# definition of a generic model


# number of components
k = 1
# size of the hidden layer in the NN
d0 = 100
# dimensionality of the data
dx = 2
# number of observations (dataset size)
N = 1000


@inf.probmodel
def nlpca(k, d0, dx, decoder):

    with inf.datamodel():
        z = inf.Normal(tf.ones([k])*0.5, 1., name="z")     # shape = [N,k]
        output = decoder(z,d0,dx)
        x_loc = output[:,:dx]
        x_scale = tf.nn.softmax(output[:,dx:])
        x = inf.Normal(x_loc, x_scale, name="x")    # shape = [N,d]


def decoder(z,d0,dx):
    h0 = tf.layers.dense(z, d0, tf.nn.relu)
    return tf.layers.dense(h0, 2 * dx)


# Q-model  approximating P

@inf.probmodel
def qmodel(k):
    with inf.datamodel():
        qz_loc = inf.Parameter(tf.ones([k])*0.5, name="qz_loc")
        qz_scale = tf.math.softplus(inf.Parameter(tf.ones([k]),name="qz_scale"))

        qz = inf.Normal(qz_loc, qz_scale, name="z")
```

(continues on next page)

(continued from previous page)

```
# create an instance of the model
m = nlpca(k,d0,dx, decoder)

# set the inference algorithm
VI = inf.inference.VI(qmodel(k), epochs=5000)

# learn the parameters
m.fit({"x": x_train}, VI)

# extract the hidden encoding
hidden_encoding = m.posterior("z").parameters()["loc"]

# project x_test into the reduced space (encode)
m.posterior("z", data={"x": x_test}).sample(5)

# sample from the posterior predictive (i.e., simulate values for x given the learnt␣
↪hidden)
m.posterior_predictive("x").sample(5)

# decode values from the hidden representation
m.posterior_predictive("x", data={"z": [2]}).sample(5)
```

7.5 Variational auto-encoder (VAE)

Similarly to the models PCA and NLPCA, a variational autoencoder allows to perform dimensionality reduction. However a VAE will contain a neural network in the P model (decoder) and another one in the Q (encoder). Its code in InferPy is shown below,

```
N = 1000

# Generate toy data
x_train = np.concatenate([
    inf.Normal([0.0, 0.0], scale=1.).sample(int(N/2)),
    inf.Normal([10.0, 10.0], scale=1.).sample(int(N/2))
    ])
x_test = np.concatenate([
    inf.Normal([0.0, 0.0], scale=1.).sample(int(N/2)),
    inf.Normal([10.0, 10.0], scale=1.).sample(int(N/2))
    ])


# number of components
k = 1
# size of the hidden layer in the NN
d0 = 100
# dimensionality of the data
dx = 2
# number of observations (dataset size)
N = 1000
```

(continues on next page)

(continued from previous page)

```
@inf.probmodel
def vae(k, d0, dx, decoder):

    with inf.datamodel():
        z = inf.Normal(tf.ones(k) * 0.5, 1., name="z")     # shape = [N,k]
        output = decoder(z, d0, dx)
        x_loc = output[:, :dx]
        x_scale = tf.nn.softmax(output[:, dx:])
        x = inf.Normal(x_loc, x_scale, name="x")    # shape = [N,d]


def decoder(z, d0, dx):    # k -> d0 -> 2*dx
    h0 = tf.layers.dense(z, d0, tf.nn.relu)
    return tf.layers.dense(h0, 2 * dx)


# Q-model  approximating P
def encoder(x, d0, k):  # dx -> d0 -> 2*k
    h0 = tf.layers.dense(x, d0, tf.nn.relu)
    return tf.layers.dense(h0, 2 * k)


@inf.probmodel
def qmodel(k, d0, dx, encoder):

    with inf.datamodel():
        x = inf.Normal(tf.ones(dx), 1, name="x")

        output = encoder(x, d0, k)
        qz_loc = output[:, :k]
        qz_scale = tf.nn.softmax(output[:, k:])

        qz = inf.Normal(qz_loc, qz_scale, name="z")


# create an instance of the model
m = vae(k, d0, dx, decoder)
```

CHAPTER

EIGHT

INFERPY PACKAGE

8.1 Subpackages

8.1.1 inferpy.contextmanager package

Submodules

inferpy.contextmanager.data_model module

inferpy.contextmanager.data_model.**datamodel**(*size=None*)
This context is used to declare a plateau model. Random Variables and Parameters will use a sample_shape defined by the argument *size*, or by the *data_model.fit*. If *size* is not specify, the default size 1, or the size specified by *fit* will be used.

inferpy.contextmanager.data_model.**fit**(*size*)

inferpy.contextmanager.data_model.**get_sample_shape**(*name*)
This function must be used inside a datamodel context (it is not checked here) If var parameters are not expanded, then expand.

name (str) The name of the variable to get its sample shape

returns a the sample_shape (number of samples of the datamodel). It is an integer, or ().

inferpy.contextmanager.data_model.**is_active**()

inferpy.contextmanager.evidence module

inferpy.contextmanager.evidence.**observe**(*variables*, *data*)

inferpy.contextmanager.randvar_registry module

inferpy.contextmanager.randvar_registry.**get_graph**()

inferpy.contextmanager.randvar_registry.**get_var_parameters**()

inferpy.contextmanager.randvar_registry.**get_variable**(*name*)

inferpy.contextmanager.randvar_registry.**get_variable_or_parameter**(*name*)

inferpy.contextmanager.randvar_registry.**init**(*graph=None*)

inferpy.contextmanager.randvar_registry.**is_building_graph**()

inferpy.contextmanager.randvar_registry.**is_default**()

inferpy.contextmanager.randvar_registry.**register_parameter**(*p*)

inferpy.contextmanager.randvar_registry.**register_variable**(*rv*)

inferpy.contextmanager.randvar_registry.**restart_default**()

inferpy.contextmanager.randvar_registry.**update_graph**(*rv_name=None*)

Module contents

8.1.2 inferpy.datasets package

Submodules

inferpy.datasets.mnist module

MNIST handwritten digits dataset.

inferpy.datasets.mnist.**load_data**(*vectorize=True, num_instances=None, num_instances_test=None, digits=[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]*)

Loads the MNIST datase

Parameters

- **vectorize** – if true, each 2D image is transformed into a 1D vector
- **num_instances** – total number of images loaded
- **digits** – list of integers indicating the digits to be considered

Returns Tuple of Numpy arrays: ‘(x_train, y_train), (x_test, y_test)

inferpy.datasets.mnist.**plot_digits**(*data, grid=[3, 3]*)

Module contents

8.1.3 inferpy.inference package

Subpackages

inferpy.inference.variational package

Subpackages

inferpy.inference.variational.loss_functions package

Submodules

inferpy.inference.variational.loss_functions.elbo module

inferpy.inference.variational.loss_functions.elbo.**ELBO**(*pvars, qvars, batch_weight=1, **kwargs*)

Compute the loss tensor from the expanded variables of p and q models. :param pvars: The dict with the

expanded p random variables :type pvars: *dict<inferpy.RandomVariable>* :param qvars: The dict with the expanded q random variables :type qvars: *dict<inferpy.RandomVariable>* :param batch_weight: Weight to assign less importance to the energy, used when processing data in batches :type batch_weight: *float*

Returns (*tf.Tensor*): The generated loss tensor

Module contents

inferpy.inference.variational.loss_functions.**ELBO**(*pvars*, *qvars*, *batch_weight=1*, ***kwargs*)

Compute the loss tensor from the expanded variables of p and q models. :param pvars: The dict with the expanded p random variables :type pvars: *dict<inferpy.RandomVariable>* :param qvars: The dict with the expanded q random variables :type qvars: *dict<inferpy.RandomVariable>* :param batch_weight: Weight to assign less importance to the energy, used when processing data in batches :type batch_weight: *float*

Returns (*tf.Tensor*): The generated loss tensor

Submodules

inferpy.inference.variational.svi module

class inferpy.inference.variational.svi.**SVI**(**args*, *batch_size=100*, ***kwargs*)

Bases: *inferpy.inference.variational.vi.VI*

compile(*pmodel*, *data_size*)

create_input_data_tensor(*sample_dict*)

update(*sample_dict*)

inferpy.inference.variational.vi module

class inferpy.inference.variational.vi.**VI**(*qmodel*, *loss='ELBO'*, *optimizer='AdamOptimizer'*, *epochs=1000*)

Bases: *inferpy.inference.inference.Inference*

compile(*pmodel*, *data_size*)

log_prob(*data*)

property losses

parameters()

sample(*size=1*, *data={}*)

update(*sample_dict*)

Module contents

Submodules

inferpy.inference.inference module

class inferpy.inference.inference.**Inference**

Bases: object

This class implements the functionality of any Inference class.

compile(*pmodel*, *data_size*)

log_prob(*data*)

parameters()

sample(*size=1*, *data={}*)

sum_log_prob(*data*)
Computes the sum of the log probabilities of a (set of) sample(s)

update(*sample_dict*)

Module contents

Any inference class must implement a run method, which receives a sample_dict object, and returns a dict of posterior objects (random distributions, list of samples, etc.)

class inferpy.inference.**SVI**(**args*, *batch_size=100*, ***kwargs*)
Bases: *inferpy.inference.variational.vi.VI*

compile(*pmodel*, *data_size*)

create_input_data_tensor(*sample_dict*)

update(*sample_dict*)

class inferpy.inference.**VI**(*qmodel*, *loss='ELBO'*, *optimizer='AdamOptimizer'*, *epochs=1000*)
Bases: *inferpy.inference.inference.Inference*

compile(*pmodel*, *data_size*)

log_prob(*data*)

property losses

parameters()

sample(*size=1*, *data={}*)

update(*sample_dict*)

8.1.4 inferpy.models package

Submodules

inferpy.models.parameter module

class inferpy.models.parameter.**Parameter**(*initial_value*, *name=None*)
Bases: object

Random Variable parameter which can be optimized by an inference mechanism.

inferpy.models.prob_model module

class inferpy.models.prob_model.**ProbModel**(*builder*)
Bases: object

Class that implements the probabilistic model functionality. It is composed of a graph, capturing the variable relationships, an OrderedDict containing the Random Variables/Parameters in order of creation, and the function which declare the Random Variables/Parameters.

expand_model(*size=1*)
Create the expanded model vars using size as plate size and return the OrderedDict

fit(*sample_dict*, *inference_method*)

plot_graph()

posterior(*target_names=None*, *data={}*)

posterior_predictive(*target_names=None*, *data={}*)

prior(*target_names=None*, *data={}*)

update(*sample_dict*)

inferpy.models.prob_model.**probmodel**(*builder*)
Decorator to create probabilistic models. The function decorated must be a function which declares the Random Variables in the model. It is not needed that the function returns such variables (we capture them using ed.tape).

inferpy.models.random_variable module

inferpy.models.random_variable.**Autoregressive**(**args*, ***kwargs*)

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for Autoregressive.

See Autoregressive for more details.

Returns RandomVariable.

Original Docstring for Distribution

Construct an *Autoregressive* distribution.

Parameters

- **distribution_fn** – Python *callable* which constructs a *tfd.Distribution*-like instance from a *Tensor* (e.g., *sample0*). The function must respect the "autoregressive property", i.e., there exists a permutation of event such that each coordinate is a diffeomorphic function of on preceding coordinates.
- **sample0** – Initial input to *distribution_fn*; used to build the distribution in *__init__* which in turn specifies this distribution's properties, e.g., *event_shape*, *batch_shape*, *dtype*. If unspecified, then *distribution_fn* should be default constructable.
- **num_steps** – Number of times *distribution_fn* is composed from samples, e.g., *num_steps=2* implies *distribution_fn(distribution_fn(sample0).sample(n)).sample()*.
- **validate_args** – Python *bool*. Whether to validate input with asserts. If *validate_args* is *False*, and the inputs are invalid, correct behavior is not guaranteed.

- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined.
- **name** – Python *str* name prefixed to Ops created by this class. Default value: "Autoregressive".

Raises

- `ValueError` – if *num_steps* and *num_elements(distribution_fn(sample0).event_shape)* are both *None*.
- `ValueError` – if *num_steps* < *1*.

`inferpy.models.random_variable.`**`BatchReshape`**`(*args, **kwargs)`

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for BatchReshape.

See BatchReshape for more details.

Returns RandomVariable.

Original Docstring for Distribution

Construct BatchReshape distribution.

Parameters

- **distribution** – The base distribution instance to reshape. Typically an instance of *Distribution*.
- **batch_shape** – Positive *int*-like vector-shaped *Tensor* representing the new shape of the batch dimensions. Up to one dimension may contain *-1*, meaning the remainder of the batch size.
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined.
- **name** – The name to give Ops created by the initializer. Default value: *"BatchReshape" + distribution.name*.

Raises

- `ValueError` – if *batch_shape* is not a vector.
- `ValueError` – if *batch_shape* has non-positive elements.
- `ValueError` – if *batch_shape* size is not the same as a *distribution.batch_shape* size.

`inferpy.models.random_variable.`**`Bernoulli`**`(*args, **kwargs)`

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for Bernoulli.

See Bernoulli for more details.

Returns RandomVariable.

Original Docstring for Distribution

Construct Bernoulli distributions.

Parameters

- **logits** – An N-D *Tensor* representing the log-odds of a *1* event. Each entry in the *Tensor* parametrizes an independent Bernoulli distribution where the probability of an event is sigmoid(logits). Only one of *logits* or *probs* should be passed in.
- **probs** – An N-D *Tensor* representing the probability of a *1* event. Each entry in the *Tensor* parameterizes an independent Bernoulli distribution. Only one of *logits* or *probs* should be passed in.
- **dtype** – The type of the event samples. Default: *int32*.
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined.
- **name** – Python *str* name prefixed to Ops created by this class.

Raises `ValueError` – If p and logits are passed, or if neither are passed.

`inferpy.models.random_variable.`**`Beta`**`(`*args*, ***kwargs*`)`

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for Beta.

See Beta for more details.

Returns RandomVariable.

Original Docstring for Distribution

Initialize a batch of Beta distributions.

Parameters

- **concentration1** – Positive floating-point *Tensor* indicating mean number of successes; aka "alpha". Implies *self.dtype* and *self.batch_shape*, i.e., *concentration1.shape = [N1, N2, ..., Nm] = self.batch_shape*.
- **concentration0** – Positive floating-point *Tensor* indicating mean number of failures; aka "beta". Otherwise has same semantics as *concentration1*.
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined.
- **name** – Python *str* name prefixed to Ops created by this class.

inferpy.models.random_variable.**Binomial**(**args*, ***kwargs*)

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for Binomial.

See Binomial for more details.

Returns RandomVariable.

Original Docstring for Distribution

Initialize a batch of Binomial distributions.

Parameters

- **total_count** – Non-negative floating point tensor with shape broadcastable to *[N1,..., Nm]* with *m >= 0* and the same dtype as *probs* or *logits*. Defines this as a batch of *N1 x ... x Nm* different Binomial distributions. Its components should be equal to integer values.
- **logits** – Floating point tensor representing the log-odds of a positive event with shape broadcastable to *[N1,..., Nm] m >= 0*, and the same dtype as *total_count*. Each entry represents logits for the probability of success for independent Binomial distributions. Only one of *logits* or *probs* should be passed in.
- **probs** – Positive floating point tensor with shape broadcastable to *[N1,..., Nm] m >= 0*, *probs in [0, 1]*. Each entry represents the probability of success for independent Binomial distributions. Only one of *logits* or *probs* should be passed in.
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined.

- **name** – Python *str* name prefixed to Ops created by this class.

inferpy.models.random_variable.**Blockwise**(**args*, ***kwargs*)

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for Blockwise.

See Blockwise for more details.

Returns RandomVariable.

Original Docstring for Distribution

Construct the *Blockwise* distribution.

Parameters

- **distributions** – Python *list* of *tfp.distributions.Distribution* instances. All distribution instances must have the same *batch_shape* and all must have *event_ndims==1*, i.e., be vector-variate distributions.
- **dtype_override** – samples of *distributions* will be cast to this *dtype*. If unspecified, all *distributions* must have the same *dtype*. Default value: *None* (i.e., do not cast).
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined.
- **name** – Python *str* name prefixed to Ops created by this class.

inferpy.models.random_variable.**Categorical**(**args*, ***kwargs*)

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for Categorical.

See Categorical for more details.

Returns RandomVariable.

Original Docstring for Distribution

Initialize Categorical distributions using class log-probabilities.

Parameters

- **logits** – An N-D *Tensor*, *N >= 1*, representing the log probabilities of a set of Categorical distributions. The first *N - 1* dimensions index into a batch of independent distributions and the last dimension represents a vector of logits for each class. Only one of *logits* or *probs* should be passed in.
- **probs** – An N-D *Tensor*, *N >= 1*, representing the probabilities of a set of Categorical distributions. The first *N - 1* dimensions index into a batch of independent distributions and the last dimension represents a vector of probabilities for each class. Only one of *logits* or *probs* should be passed in.
- **dtype** – The type of the event samples (default: int32).
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined.
- **name** – Python *str* name prefixed to Ops created by this class.

inferpy.models.random_variable.**Cauchy**(**args*, ***kwargs*)

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for Cauchy.

See Cauchy for more details.

Returns RandomVariable.

Original Docstring for Distribution

Construct Cauchy distributions.

The parameters *loc* and *scale* must be shaped in a way that supports broadcasting (e.g. *loc + scale* is a valid operation).

Parameters

- **loc** – Floating point tensor; the modes of the distribution(s).
- **scale** – Floating point tensor; the locations of the distribution(s). Must contain only positive values.
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined.

- **name** – Python *str* name prefixed to Ops created by this class.

Raises TypeError – if *loc* and *scale* have different *dtype*.

inferpy.models.random_variable.**Chi**(**args*, ***kwargs*)

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for Chi.

See Chi for more details.

Returns RandomVariable.

Original Docstring for Distribution

Construct Chi distributions with parameter *df*.

Parameters

- **df** – Floating point tensor, the degrees of freedom of the distribution(s). *df* must contain only positive values.
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value *NaN* to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined.
- **name** – Python *str* name prefixed to Ops created by this class. Default value: *'Chi'*.

inferpy.models.random_variable.**Chi2**(**args*, ***kwargs*)

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for Chi2.

See Chi2 for more details.

Returns RandomVariable.

Original Docstring for Distribution

Construct Chi2 distributions with parameter *df*.

Parameters

- **df** – Floating point tensor, the degrees of freedom of the distribution(s). *df* must contain only positive values.
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined.
- **name** – Python *str* name prefixed to Ops created by this class.

inferpy.models.random_variable.**Chi2WithAbsDf**(**args*, ***kwargs*)

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for Chi2WithAbsDf.

See Chi2WithAbsDf for more details.

Returns RandomVariable.

Original Docstring for Distribution

DEPRECATED FUNCTION

Warning: THIS FUNCTION IS DEPRECATED. It will be removed after 2019-06-05. Instructions for updating: Chi2WithAbsDf is deprecated, use Chi2(df=tf.floor(tf.abs(df))) instead.

inferpy.models.random_variable.**ConditionalDistribution**(**args*, ***kwargs*)

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for ConditionalDistribution.

See ConditionalDistribution for more details.

Returns RandomVariable.

Original Docstring for Distribution

Constructs the *Distribution*.

This is a private method for subclass use.

Parameters

- **dtype** – The type of the event samples. *None* implies no type-enforcement.

- **reparameterization_type** – Instance of *ReparameterizationType*. If *tfd.FULLY_REPARAMETERIZED*, this *Distribution* can be reparameterized in terms of some standard distribution with a function whose Jacobian is constant for the support of the standard distribution. If *tfd.NOT_REPARAMETERIZED*, then no such reparameterization is available.
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined.
- **parameters** – Python *dict* of parameters used to instantiate this *Distribution*.
- **graph_parents** – Python *list* of graph prerequisites of this *Distribution*.
- **name** – Python *str* name prefixed to Ops created by this class. Default: subclass name.

Raises `ValueError` – if any member of graph_parents is *None* or not a *Tensor*.

`inferpy.models.random_variable.`**`ConditionalTransformedDistribution`**`(`**args*, ***kwargs*`)`

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for ConditionalTransformedDistribution.

See ConditionalTransformedDistribution for more details.

Returns RandomVariable.

Original Docstring for Distribution

Construct a Transformed Distribution.

Parameters

- **distribution** – The base distribution instance to transform. Typically an instance of *Distribution*.
- **bijector** – The object responsible for calculating the transformation. Typically an instance of *Bijector*.
- **batch_shape** – *integer* vector *Tensor* which overrides *distribution batch_shape*; valid only if *distribution.is_scalar_batch()*.
- **event_shape** – *integer* vector *Tensor* which overrides *distribution event_shape*; valid only if *distribution.is_scalar_event()*.
- **kwargs_split_fn** – Python *callable* which takes a kwargs *dict* and returns a tuple of kwargs *dict's for each of the 'distribution* and *bijector* parameters respectively. Default value: *_default_kwargs_split_fn* (i.e.,

  'lambda kwargs: (kwargs.get('distribution_kwargs', {}),
  kwargs.get('bijector_kwargs', {}))')

- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **parameters** – Locals dict captured by subclass constructor, to be used for copy/slice re-instantiation operations.
- **name** – Python *str* name prefixed to Ops created by this class. Default: *bijector.name* + *distribution.name*.

inferpy.models.random_variable.**Deterministic**(**args*, ***kwargs*)

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for Deterministic.

See Deterministic for more details.

Returns RandomVariable.

Original Docstring for Distribution

Initialize a scalar *Deterministic* distribution.

The *atol* and *rtol* parameters allow for some slack in *pmf*, *cdf* computations, e.g. due to floating-point error.

‘‘‘ pmf(x; loc)

= 1, if Abs(x - loc) <= atol + rtol * Abs(loc), = 0, otherwise.

‘‘‘

Parameters

- **loc** – Numeric *Tensor* of shape *[B1, ..., Bb]*, with *b >= 0*. The point (or batch of points) on which this distribution is supported.
- **atol** – Non-negative *Tensor* of same *dtype* as *loc* and broadcastable shape. The absolute tolerance for comparing closeness to *loc*. Default is *0*.
- **rtol** – Non-negative *Tensor* of same *dtype* as *loc* and broadcastable shape. The relative tolerance for comparing closeness to *loc*. Default is *0*.
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value “*NaN*” to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic’s batch members are undefined.
- **name** – Python *str* name prefixed to Ops created by this class.

inferpy.models.random_variable.**Dirichlet**(**args*, ***kwargs*)

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for Dirichlet.

See Dirichlet for more details.

Returns RandomVariable.

Original Docstring for Distribution

Initialize a batch of Dirichlet distributions.

Parameters

- **concentration** – Positive floating-point *Tensor* indicating mean number of class occurrences; aka "alpha". Implies *self.dtype*, and *self.batch_shape*, *self.event_shape*, i.e., if *concentration.shape = [N1, N2, ..., Nm, k]* then *batch_shape = [N1, N2, ..., Nm]* and *event_shape = [k]*.
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined.
- **name** – Python *str* name prefixed to Ops created by this class.

inferpy.models.random_variable.**DirichletMultinomial**(**args*, ***kwargs*)

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for DirichletMultinomial.

See DirichletMultinomial for more details.

Returns RandomVariable.

Original Docstring for Distribution

Initialize a batch of DirichletMultinomial distributions.

Parameters

- **total_count** – Non-negative floating point tensor, whose dtype is the same as *concentration*. The shape is broadcastable to *[N1,..., Nm]* with *m >= 0*. Defines this as a batch of *N1 x ... x Nm* different Dirichlet multinomial distributions. Its components should be equal to integer values.

- **concentration** – Positive floating point tensor, whose dtype is the same as *n* with shape broadcastable to *[N1,..., Nm, K] m >= 0*. Defines this as a batch of *N1 x ... x Nm* different *K* class Dirichlet multinomial distributions.
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined.
- **name** – Python *str* name prefixed to Ops created by this class.

inferpy.models.random_variable.**Distribution**(**args*, ***kwargs*)

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for Distribution.

See Distribution for more details.

Returns RandomVariable.

Original Docstring for Distribution

Constructs the *Distribution*.

This is a private method for subclass use.

Parameters

- **dtype** – The type of the event samples. *None* implies no type-enforcement.
- **reparameterization_type** – Instance of *ReparameterizationType*. If *tfd.FULLY_REPARAMETERIZED*, this *Distribution* can be reparameterized in terms of some standard distribution with a function whose Jacobian is constant for the support of the standard distribution. If *tfd.NOT_REPARAMETERIZED*, then no such reparameterization is available.
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined.
- **parameters** – Python *dict* of parameters used to instantiate this *Distribution*.
- **graph_parents** – Python *list* of graph prerequisites of this *Distribution*.
- **name** – Python *str* name prefixed to Ops created by this class. Default: subclass name.

Raises `ValueError` – if any member of graph_parents is *None* or not a *Tensor*.

inferpy.models.random_variable.**Empirical**(**args*, ***kwargs*)

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for Empirical.

See Empirical for more details.

Returns RandomVariable.

Original Docstring for Distribution

Initialize *Empirical* distributions.

Parameters

- **samples** – Numeric *Tensor* of shape [B1, ..., Bk, S, E1, ..., En]', *k, n >= 0*. Samples or batches of samples on which the distribution is based. The first *k* dimensions index into a batch of independent distributions. Length of *S* dimension determines number of samples in each multiset. The last *n* dimension represents samples for each distribution. n is specified by argument event_ndims.
- **event_ndims** – Python *int32*, default *0*. number of dimensions for each event. When *0* this distribution has scalar samples. When *1* this distribution has vector-like samples.
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value *NaN* to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined.
- **name** – Python *str* name prefixed to Ops created by this class.

Raises `ValueError` – if the rank of *samples* < event_ndims + 1.

`inferpy.models.random_variable.`**`ExpRelaxedOneHotCategorical`**`(*args, **kwargs)`

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for ExpRelaxedOneHotCategorical.

See ExpRelaxedOneHotCategorical for more details.

Returns RandomVariable.

Original Docstring for Distribution

Initialize ExpRelaxedOneHotCategorical using class log-probabilities.

Parameters

- **temperature** – An 0-D *Tensor*, representing the temperature of a set of ExpRelaxedCategorical distributions. The temperature should be positive.
- **logits** – An N-D *Tensor*, *N >= 1*, representing the log probabilities of a set of ExpRelaxedCategorical distributions. The first *N - 1* dimensions index into a batch of independent distributions and the last dimension represents a vector of logits for each class. Only one of *logits* or *probs* should be passed in.
- **probs** – An N-D *Tensor*, *N >= 1*, representing the probabilities of a set of ExpRelaxedCategorical distributions. The first *N - 1* dimensions index into a batch of independent distributions and the last dimension represents a vector of probabilities for each class. Only one of *logits* or *probs* should be passed in.
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined.
- **name** – Python *str* name prefixed to Ops created by this class.

inferpy.models.random_variable.**Exponential**(**args*, ***kwargs*)

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for Exponential.

See Exponential for more details.

Returns RandomVariable.

Original Docstring for Distribution

Construct Exponential distribution with parameter *rate*.

Parameters

- **rate** – Floating point tensor, equivalent to *1 / mean*. Must contain only positive values.
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined.
- **name** – Python *str* name prefixed to Ops created by this class.

inferpy.models.random_variable.**FiniteDiscrete**(**args*, ***kwargs*)

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for FiniteDiscrete.

See FiniteDiscrete for more details.

Returns RandomVariable.

Original Docstring for Distribution

Construct a finite discrete contribution.

Parameters

- **outcomes** – A 1-D floating or integer *Tensor*, representing a list of possible outcomes in strictly ascending order.
- **logits** – A floating N-D *Tensor*, *N >= 1*, representing the log probabilities of a set of FiniteDiscrete distributions. The first *N - 1* dimensions index into a batch of independent distributions and the last dimension represents a vector of logits for each discrete value. Only one of *logits* or *probs* should be passed in.
- **probs** – A floating N-D *Tensor*, *N >= 1*, representing the probabilities of a set of FiniteDiscrete distributions. The first *N - 1* dimensions index into a batch of independent distributions and the last dimension represents a vector of probabilities for each discrete value. Only one of *logits* or *probs* should be passed in.
- **rtol** – *Tensor* with same *dtype* as *outcomes*. The relative tolerance for floating number comparison. Only effective when *outcomes* is a floating *Tensor*. Default is *10 * eps*.
- **atol** – *Tensor* with same *dtype* as *outcomes*. The absolute tolerance for floating number comparison. Only effective when *outcomes* is a floating *Tensor*. Default is *10 * eps*.
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may render incorrect outputs.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value '*NaN*' to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined.
- **name** – Python *str* name prefixed to Ops created by this class.

`inferpy.models.random_variable.`**`Gamma`**`(*args, **kwargs)`

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for Gamma.

See Gamma for more details.

Returns RandomVariable.

Original Docstring for Distribution

Construct Gamma with *concentration* and *rate* parameters.

The parameters *concentration* and *rate* must be shaped in a way that supports broadcasting (e.g. *concentration + rate* is a valid operation).

Parameters

- **concentration** – Floating point tensor, the concentration params of the distribution(s). Must contain only positive values.
- **rate** – Floating point tensor, the inverse scale params of the distribution(s). Must contain only positive values.
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined.
- **name** – Python *str* name prefixed to Ops created by this class.

Raises `TypeError` – if *concentration* and *rate* are different dtypes.

`inferpy.models.random_variable.`**GammaGamma**(**args*, ***kwargs*)

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for GammaGamma.

See GammaGamma for more details.

Returns RandomVariable.

Original Docstring for Distribution

Initializes a batch of Gamma-Gamma distributions.

The parameters *concentration* and *rate* must be shaped in a way that supports broadcasting (e.g. *concentration + mixing_concentration + mixing_rate* is a valid operation).

Parameters

- **concentration** – Floating point tensor, the concentration params of the distribution(s). Must contain only positive values.
- **mixing_concentration** – Floating point tensor, the concentration params of the mixing Gamma distribution(s). Must contain only positive values.

- **mixing_rate** – Floating point tensor, the rate params of the mixing Gamma distribution(s). Must contain only positive values.
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined.
- **name** – Python *str* name prefixed to Ops created by this class.

Raises `TypeError` – if *concentration* and *rate* are different dtypes.

`inferpy.models.random_variable.`**`GaussianProcess`**`(*args, **kwargs)`

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for GaussianProcess.

See GaussianProcess for more details.

Returns RandomVariable.

Original Docstring for Distribution

Instantiate a GaussianProcess Distribution.

Parameters

- **kernel** – *PositiveSemidefiniteKernel*-like instance representing the GP's covariance function.
- **index_points** – *float Tensor* representing finite (batch of) vector(s) of points in the index set over which the GP is defined. Shape has the form *[b1, ..., bB, e, f1, ..., fF]* where *F* is the number of feature dimensions and must equal *kernel.feature_ndims* and *e* is the number (size) of index points in each batch. Ultimately this distribution corresponds to a *e*-dimensional multivariate normal. The batch shape must be broadcastable with *kernel.batch_shape* and any batch dims yielded by *mean_fn*.
- **mean_fn** – Python *callable* that acts on *index_points* to produce a (batch of) vector(s) of mean values at *index_points*. Takes a *Tensor* of shape *[b1, ..., bB, f1, ..., fF]* and returns a *Tensor* whose shape is broadcastable with *[b1, ..., bB]*. Default value: *None* implies constant zero function.
- **observation_noise_variance** – *float Tensor* representing the variance of the noise in the Normal likelihood distribution of the model. May be batched, in which case the batch shape must be broadcastable with the shapes of all other batched parameters (*kernel.batch_shape*, *index_points*, etc.). Default value: *0.*
- **jitter** – *float* scalar *Tensor* added to the diagonal of the covariance matrix to ensure positive definiteness of the covariance matrix. Default value: *1e-6*.

- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs. Default value: *False*.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined. Default value: *False*.
- **name** – Python *str* name prefixed to Ops created by this class. Default value: "GaussianProcess".

Raises `ValueError` – if *mean_fn* is not *None* and is not callable.

`inferpy.models.random_variable.`**`GaussianProcessRegressionModel`**`(*args, **kwargs)`

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for GaussianProcessRegressionModel.

See GaussianProcessRegressionModel for more details.

Returns RandomVariable.

Original Docstring for Distribution

Construct a GaussianProcessRegressionModel instance.

Parameters

- **kernel** – *PositiveSemidefiniteKernel*-like instance representing the GP's covariance function.
- **index_points** – *float Tensor* representing finite collection, or batch of collections, of points in the index set over which the GP is defined. Shape has the form *[b1, ..., bB, e, f1, ..., fF]* where *F* is the number of feature dimensions and must equal *kernel.feature_ndims* and *e* is the number (size) of index points in each batch. Ultimately this distribution corresponds to an *e*-dimensional multivariate normal. The batch shape must be broadcastable with *kernel.batch_shape* and any batch dims yielded by *mean_fn*.
- **observation_index_points** – *float Tensor* representing finite collection, or batch of collections, of points in the index set for which some data has been observed. Shape has the form *[b1, ..., bB, e, f1, ..., fF]* where *F* is the number of feature dimensions and must equal *kernel.feature_ndims*, and *e* is the number (size) of index points in each batch. *[b1, ..., bB, e]* must be broadcastable with the shape of *observations*, and *[b1, ..., bB]* must be broadcastable with the shapes of all other batched parameters (*kernel.batch_shape*, *index_points*, etc). The default value is *None*, which corresponds to the empty set of observations, and simply results in the prior predictive model (a GP with noise of variance *predictive_noise_variance*).
- **observations** – *float Tensor* representing collection, or batch of collections, of observations corresponding to *observation_index_points*. Shape has the form *[b1, ..., bB, e]*, which must be brodcastable with the batch and example shapes of *observation_index_points*. The batch shape *[b1, ..., bB]* must be broadcastable with the shapes of all other batched parameters (*kernel.batch_shape*, *index_points*, etc.). The default value is *None*, which corresponds

to the empty set of observations, and simply results in the prior predictive model (a GP with noise of variance *predictive_noise_variance*).

- **observation_noise_variance** – *float Tensor* representing the variance of the noise in the Normal likelihood distribution of the model. May be batched, in which case the batch shape must be broadcastable with the shapes of all other batched parameters (*kernel.batch_shape*, *index_points*, etc.). Default value: *0.*
- **predictive_noise_variance** – *float Tensor* representing the variance in the posterior predictive model. If *None*, we simply re-use *observation_noise_variance* for the posterior predictive noise. If set explicitly, however, we use this value. This allows us, for example, to omit predictive noise variance (by setting this to zero) to obtain noiseless posterior predictions of function values, conditioned on noisy observations.
- **mean_fn** – Python *callable* that acts on *index_points* to produce a collection, or batch of collections, of mean values at *index_points*. Takes a *Tensor* of shape *[b1, ..., bB, f1, ..., fF]* and returns a *Tensor* whose shape is broadcastable with *[b1, ..., bB]*. Default value: *None* implies the constant zero function.
- **jitter** – *float* scalar *Tensor* added to the diagonal of the covariance matrix to ensure positive definiteness of the covariance matrix. Default value: *1e-6*.
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs. Default value: *False*.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value *NaN* to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined. Default value: *False*.
- **name** – Python *str* name prefixed to Ops created by this class. Default value: 'GaussianProcessRegressionModel'.

Raises `ValueError` – if either - only one of *observations* and *observation_index_points* is given, or - *mean_fn* is not *None* and not callable.

`inferpy.models.random_variable.`**`Geometric`**`(*args, **kwargs)`

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for Geometric.

See Geometric for more details.

Returns RandomVariable.

Original Docstring for Distribution

Construct Geometric distributions.

Parameters

- **logits** – Floating-point *Tensor* with shape *[B1, ..., Bb]* where *b >= 0* indicates the number of batch dimensions. Each entry represents logits for the probability of success for

independent Geometric distributions and must be in the range *(-inf, inf]*. Only one of *logits* or *probs* should be specified.

- **probs** – Positive floating-point *Tensor* with shape *[B1, ..., Bb]* where *b >= 0* indicates the number of batch dimensions. Each entry represents the probability of success for independent Geometric distributions and must be in the range *(0, 1]*. Only one of *logits* or *probs* should be specified.
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined.
- **name** – Python *str* name prefixed to Ops created by this class.

inferpy.models.random_variable.**Gumbel**(**args*, ***kwargs*)

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for Gumbel.

See Gumbel for more details.

Returns RandomVariable.

Original Docstring for Distribution

Construct Gumbel distributions with location and scale *loc* and *scale*.

The parameters *loc* and *scale* must be shaped in a way that supports broadcasting (e.g. *loc + scale* is a valid operation).

Parameters

- **loc** – Floating point tensor, the means of the distribution(s).
- **scale** – Floating point tensor, the scales of the distribution(s). scale must contain only positive values.
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs. Default value: *False*.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined. Default value: *True*.
- **name** – Python *str* name prefixed to Ops created by this class. Default value: *'Gumbel'*.

Raises TypeError – if loc and scale are different dtypes.

inferpy.models.random_variable.**HalfCauchy**(**args*, ***kwargs*)

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for HalfCauchy.

See HalfCauchy for more details.

Returns RandomVariable.

Original Docstring for Distribution

Construct a half-Cauchy distribution with *loc* and *scale*.

Parameters

- **loc** – Floating-point *Tensor*; the location(s) of the distribution(s).
- **scale** – Floating-point *Tensor*; the scale(s) of the distribution(s). Must contain only positive values.
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs. Default value: *False* (i.e. do not validate args).
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined. Default value: *True*.
- **name** – Python *str* name prefixed to Ops created by this class. Default value: 'HalfCauchy'.

Raises `TypeError` – if *loc* and *scale* have different *dtype*.

`inferpy.models.random_variable.`**`HalfNormal`**`(`**args*, ***kwargs*`)`

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for HalfNormal.

See HalfNormal for more details.

Returns RandomVariable.

Original Docstring for Distribution

Construct HalfNormals with scale *scale*.

Parameters

- **scale** – Floating point tensor; the scales of the distribution(s). Must contain only positive values.

- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined.
- **name** – Python *str* name prefixed to Ops created by this class.

inferpy.models.random_variable.**HiddenMarkovModel**(**args*, ***kwargs*)

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for HiddenMarkovModel.

See HiddenMarkovModel for more details.

Returns RandomVariable.

Original Docstring for Distribution

Initialize hidden Markov model.

Parameters

- **initial_distribution** – A *Categorical*-like instance. Determines probability of first hidden state in Markov chain. The number of categories must match the number of categories of *transition_distribution* as well as both the rightmost batch dimension of *transition_distribution* and the rightmost batch dimension of *observation_distribution*.
- **transition_distribution** – A *Categorical*-like instance. The rightmost batch dimension indexes the probability distribution of each hidden state conditioned on the previous hidden state.
- **observation_distribution** – A *tfp.distributions.Distribution*-like instance. The rightmost batch dimension indexes the distribution of each observation conditioned on the corresponding hidden state.
- **num_steps** – The number of steps taken in Markov chain. A python *int*.
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs. Default value: *False*.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined. Default value: *True*.
- **name** – Python *str* name prefixed to Ops created by this class. Default value: "HiddenMarkovModel".

Raises

- ValueError – if *num_steps* is not at least 1.

- `ValueError` – if *initial_distribution* does not have scalar *event_shape*.
- `ValueError` – if *transition_distribution* does not have scalar *event_shape.*
- `ValueError` – if *transition_distribution* and *observation_distribution* are fully defined but don't have matching rightmost dimension.

`inferpy.models.random_variable.`**`Horseshoe`**`(*args, **kwargs)`

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for Horseshoe.

See Horseshoe for more details.

Returns RandomVariable.

Original Docstring for Distribution

Construct a Horseshoe distribution with *scale*.

Parameters

- **`scale`** – Floating point tensor; the scales of the distribution(s). Must contain only positive values.
- **`validate_args`** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs. Default value: *False* (i.e., do not validate args).
- **`allow_nan_stats`** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined. Default value: *True*.
- **`name`** – Python *str* name prefixed to Ops created by this class. Default value: 'Horseshoe'.

`inferpy.models.random_variable.`**`Independent`**`(*args, **kwargs)`

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for Independent.

See Independent for more details.

Returns RandomVariable.

Original Docstring for Distribution

Construct a *Independent* distribution.

Parameters

- **distribution** – The base distribution instance to transform. Typically an instance of *Distribution*.
- **reinterpreted_batch_ndims** – Scalar, integer number of rightmost batch dims which will be regarded as event dims. When *None* all but the first batch axis (batch axis 0) will be transferred to event dimensions (analogous to *tf.layers.flatten*).
- **validate_args** – Python *bool*. Whether to validate input with asserts. If *validate_args* is *False*, and the inputs are invalid, correct behavior is not guaranteed.
- **name** – The name for ops managed by the distribution. Default value: *Independent* + *distribution.name*.

Raises `ValueError` – if *reinterpreted_batch_ndims* exceeds *distribution.batch_ndims*

`inferpy.models.random_variable.`**`InverseGamma`**`(`**args*, ***kwargs*`)`

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for InverseGamma.

See InverseGamma for more details.

Returns RandomVariable.

Original Docstring for Distribution

Construct InverseGamma with *concentration* and *scale* parameters. (deprecated arguments)

Warning: SOME ARGUMENTS ARE DEPRECATED: *(rate)*. They will be removed after 2019-05-08. Instructions for updating: The *rate* parameter is deprecated. Use *scale* instead.The *rate* parameter was always interpreted as a *scale* parameter, but erroneously misnamed.

The parameters *concentration* and *scale* must be shaped in a way that supports broadcasting (e.g. *concentration* + *scale* is a valid operation).

Parameters

- **concentration** – Floating point tensor, the concentration params of the distribution(s). Must contain only positive values.
- **scale** – Floating point tensor, the scale params of the distribution(s). Must contain only positive values.
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined.
- **rate** – Deprecated (mis-named) alias for *scale*.
- **name** – Python *str* name prefixed to Ops created by this class.

Raises `TypeError` – if *concentration* and *scale* are different dtypes.

`inferpy.models.random_variable.`**`InverseGaussian`**`(*args, **kwargs)`

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for InverseGaussian.

See InverseGaussian for more details.

Returns RandomVariable.

Original Docstring for Distribution

Constructs inverse Gaussian distribution with *loc* and *concentration*.

Parameters

- **`loc`** – Floating-point *Tensor*, the loc params. Must contain only positive values.
- **`concentration`** – Floating-point *Tensor*, the concentration params. Must contain only positive values.
- **`validate_args`** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs. Default value: *False* (i.e. do not validate args).
- **`allow_nan_stats`** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined. Default value: *True*.
- **`name`** – Python *str* name prefixed to Ops created by this class. Default value: 'InverseGaussian'.

`inferpy.models.random_variable.`**`JointDistribution`**`(*args, **kwargs)`

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for JointDistribution.

See JointDistribution for more details.

Returns RandomVariable.

Original Docstring for Distribution

Constructs the *Distribution*.

This is a private method for subclass use.

Parameters

- **dtype** – The type of the event samples. *None* implies no type-enforcement.
- **reparameterization_type** – Instance of *ReparameterizationType*. If *tfd.FULLY_REPARAMETERIZED*, this *Distribution* can be reparameterized in terms of some standard distribution with a function whose Jacobian is constant for the support of the standard distribution. If *tfd.NOT_REPARAMETERIZED*, then no such reparameterization is available.
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined.
- **parameters** – Python *dict* of parameters used to instantiate this *Distribution*.
- **graph_parents** – Python *list* of graph prerequisites of this *Distribution*.
- **name** – Python *str* name prefixed to Ops created by this class. Default: subclass name.

Raises `ValueError` – if any member of graph_parents is *None* or not a *Tensor*.

`inferpy.models.random_variable.`**`JointDistributionCoroutine`**`(`**args*, ***kwargs*`)`

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for JointDistributionCoroutine.

See JointDistributionCoroutine for more details.

Returns RandomVariable.

Original Docstring for Distribution

Construct the *JointDistributionCoroutine* distribution.

Parameters

- **model** – A generator that yields a sequence of *tfd.Distribution*-like instances.
- **sample_dtype** – Samples from this distribution will be structured like *tf.nest.pack_sequence_as(sample_dtype, list_)*. *sample_dtype* is only used for *tf.nest.pack_sequence_as* structuring of outputs, never casting (which is the responsibility of the component distributions). Default value: *None* (i.e., *tuple*).
- **validate_args** – Python *bool*. Whether to validate input with asserts. If *validate_args* is *False*, and the inputs are invalid, correct behavior is not guaranteed. Default value: *False*.
- **name** – The name for ops managed by the distribution. Default value: *None* (i.e., *"JointDistributionCoroutine"*).

`inferpy.models.random_variable.`**`JointDistributionNamed`**`(`**args*, ***kwargs*`)`

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for JointDistributionNamed.

See JointDistributionNamed for more details.

Returns RandomVariable.

Original Docstring for Distribution

Construct the *JointDistributionNamed* distribution.

Parameters

- **model** – Python *dict* or *namedtuple* of distribution-making functions each with required args corresponding only to other keys.
- **validate_args** – Python *bool*. Whether to validate input with asserts. If *validate_args* is *False*, and the inputs are invalid, correct behavior is not guaranteed. Default value: *False*.
- **name** – The name for ops managed by the distribution. Default value: *None* (i.e., *"JointDistributionNamed"*).

inferpy.models.random_variable.**JointDistributionSequential**(**args*, ***kwargs*)

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for JointDistributionSequential.

See JointDistributionSequential for more details.

Returns RandomVariable.

Original Docstring for Distribution

Construct the *JointDistributionSequential* distribution.

Parameters

- **model** – Python list of either tfd.Distribution instances and/or lambda functions which take the *k* previous distributions and returns a new tfd.Distribution instance.
- **validate_args** – Python *bool*. Whether to validate input with asserts. If *validate_args* is *False*, and the inputs are invalid, correct behavior is not guaranteed. Default value: *False*.
- **name** – The name for ops managed by the distribution. Default value: *None* (i.e., *"JointDistributionSequential"*).

class inferpy.models.random_variable.**Kind**
Bases: enum.IntEnum

An enumeration.

GLOBAL_HIDDEN = 0

GLOBAL_OBSERVED = 1

LOCAL_HIDDEN = 2

LOCAL_OBSERVED = 3

inferpy.models.random_variable.**Kumaraswamy**(**args*, ***kwargs*)

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for Kumaraswamy.

See Kumaraswamy for more details.

Returns RandomVariable.

Original Docstring for Distribution

Initialize a batch of Kumaraswamy distributions.

Parameters

- **concentration1** – Positive floating-point *Tensor* indicating mean number of successes; aka "alpha". Implies *self.dtype* and *self.batch_shape*, i.e., *concentration1.shape = [N1, N2, ..., Nm] = self.batch_shape*.
- **concentration0** – Positive floating-point *Tensor* indicating mean number of failures; aka "beta". Otherwise has same semantics as *concentration1*.
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined.
- **name** – Python *str* name prefixed to Ops created by this class.

inferpy.models.random_variable.**LKJ**(**args*, ***kwargs*)

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for LKJ.

See LKJ for more details.

Returns RandomVariable.

Original Docstring for Distribution

Construct LKJ distributions.

Parameters

- **dimension** – Python *int*. The dimension of the correlation matrices to sample.
- **concentration** – *float* or *double Tensor*. The positive concentration parameter of the LKJ distributions. The pdf of a sample matrix *X* is proportional to *det(X) ** (concentration - 1)*.
- **input_output_cholesky** – Python *bool*. If *True*, functions whose input or output have the semantics of samples assume inputs are in Cholesky form and return outputs in Cholesky form. In particular, if this flag is *True*, input to *log_prob* is presumed of Cholesky form and output from *sample* is of Cholesky form. Setting this argument to *True* is purely a computational optimization and does not change the underlying distribution. Additionally, validation checks which are only defined on the multiplied-out form are omitted, even if *validate_args* is *True*. Default value: *False* (i.e., input/output does not have Cholesky semantics).
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value *NaN* to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined.
- **name** – Python *str* name prefixed to Ops created by this class.

Raises `ValueError` – If *dimension* is negative.

`inferpy.models.random_variable.`**`Laplace`**`(`**args*, ***kwargs*`)`

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for Laplace.

See Laplace for more details.

Returns RandomVariable.

Original Docstring for Distribution

Construct Laplace distribution with parameters *loc* and *scale*.

The parameters *loc* and *scale* must be shaped in a way that supports broadcasting (e.g., *loc / scale* is a valid operation).

Parameters

- **loc** – Floating point tensor which characterizes the location (center) of the distribution.
- **scale** – Positive floating point tensor which characterizes the spread of the distribution.
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined.
- **name** – Python *str* name prefixed to Ops created by this class.

Raises `TypeError` – if *loc* and *scale* are of different dtype.

`inferpy.models.random_variable.`**`LinearGaussianStateSpaceModel`**`(*args, **kwargs)`

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for LinearGaussianStateSpaceModel.

See LinearGaussianStateSpaceModel for more details.

Returns RandomVariable.

Original Docstring for Distribution

Initialize a 'LinearGaussianStateSpaceModel.

Parameters

- **num_timesteps** – Integer *Tensor* total number of timesteps.
- **transition_matrix** – A transition operator, represented by a Tensor or LinearOperator of shape *[latent_size, latent_size]*, or by a callable taking as argument a scalar integer Tensor *t* and returning a Tensor or LinearOperator representing the transition operator from latent state at time *t* to time *t + 1*.
- **transition_noise** – An instance of *tfd.MultivariateNormalLinearOperator* with event shape *[latent_size]*, representing the mean and covariance of the transition noise model, or a callable taking as argument a scalar integer Tensor *t* and returning such a distribution representing the noise in the transition from time *t* to time *t + 1*.
- **observation_matrix** – An observation operator, represented by a Tensor or LinearOperator of shape *[observation_size, latent_size]*, or by a callable taking as argument a scalar integer Tensor *t* and returning a timestep-specific Tensor or LinearOperator.
- **observation_noise** – An instance of *tfd.MultivariateNormalLinearOperator* with event shape *[observation_size]*, representing the mean and covariance of the observation noise model, or a callable taking as argument a scalar integer Tensor *t* and returning a timestep-specific noise model.
- **initial_state_prior** – An instance of *MultivariateNormalLinearOperator* representing the prior distribution on latent states; must have event shape *[latent_size]*.

- **initial_step** – optional *int* specifying the time of the first modeled timestep. This is added as an offset when passing timesteps *t* to (optional) callables specifying timestep-specific transition and observation models.
- **validate_args** – Python *bool*, default *False*. Whether to validate input with asserts. If *validate_args* is *False*, and the inputs are invalid, correct behavior is not guaranteed.
- **allow_nan_stats** – Python *bool*, default *True*. If *False*, raise an exception if a statistic (e.g. mean/mode/etc...) is undefined for any batch member If *True*, batch members with valid parameters leading to undefined statistics will return NaN for this statistic.
- **name** – The name to give Ops created by the initializer.

inferpy.models.random_variable.**LogNormal**(**args*, ***kwargs*)

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for LogNormal.

See LogNormal for more details.

Returns RandomVariable.

Original Docstring for Distribution

Construct a log-normal distribution.

The LogNormal distribution models positive-valued random variables whose logarithm is normally distributed with mean *loc* and standard deviation *scale*. It is constructed as the exponential transformation of a Normal distribution.

Parameters

- **loc** – Floating-point *Tensor*; the means of the underlying Normal distribution(s).
- **scale** – Floating-point *Tensor*; the stddevs of the underlying Normal distribution(s).
- **validate_args** – Python *bool*, default *False*. Whether to validate input with asserts. If *validate_args* is *False*, and the inputs are invalid, correct behavior is not guaranteed.
- **allow_nan_stats** – Python *bool*, default *True*. If *False*, raise an exception if a statistic (e.g. mean/mode/etc...) is undefined for any batch member If *True*, batch members with valid parameters leading to undefined statistics will return NaN for this statistic.
- **name** – The name to give Ops created by the initializer.

inferpy.models.random_variable.**Logistic**(**args*, ***kwargs*)

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for Logistic.

See Logistic for more details.

Returns RandomVariable.

Original Docstring for Distribution

Construct Logistic distributions with mean and scale *loc* and *scale*.

The parameters *loc* and *scale* must be shaped in a way that supports broadcasting (e.g. *loc + scale* is a valid operation).

Parameters

- **loc** – Floating point tensor, the means of the distribution(s).
- **scale** – Floating point tensor, the scales of the distribution(s). Must contain only positive values.
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined.
- **name** – The name to give Ops created by the initializer.

Raises `TypeError` – if loc and scale are different dtypes.

`inferpy.models.random_variable.`**`Mixture`**`(*args, **kwargs)`

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for Mixture.

See Mixture for more details.

Returns RandomVariable.

Original Docstring for Distribution

Initialize a Mixture distribution.

A *Mixture* is defined by a *Categorical* (*cat*, representing the mixture probabilities) and a list of *Distribution* objects all having matching dtype, batch shape, event shape, and continuity properties (the components).

The *num_classes* of *cat* must be possible to infer at graph construction time and match *len(components)*.

Parameters

- **cat** – A *Categorical* distribution instance, representing the probabilities of *distributions*.
- **components** – A list or tuple of *Distribution* instances. Each instance must have the same type, be defined on the same domain, and have matching *event_shape* and *batch_shape*.

- **validate_args** – Python *bool*, default *False*. If *True*, raise a runtime error if batch or event ranks are inconsistent between cat and any of the distributions. This is only checked if the ranks cannot be determined statically at graph construction time.
- **allow_nan_stats** – Boolean, default *True*. If *False*, raise an exception if a statistic (e.g. mean/mode/etc...) is undefined for any

  batch member. If *True*, batch members with valid parameters leading to undefined statistics will return NaN for this statistic.
- **use_static_graph** – Calls to *sample* will not rely on dynamic tensor indexing, allowing for some static graph compilation optimizations, but at the expense of sampling all underlying distributions in the mixture. (Possibly useful when running on TPUs). Default value: *False* (i.e., use dynamic indexing).
- **name** – A name for this distribution (optional).

Raises

- `TypeError` – If cat is not a *Categorical*, or *components* is not a list or tuple, or the elements of *components* are not instances of *Distribution*, or do not have matching *dtype*.
- `ValueError` – If *components* is an empty list or tuple, or its elements do not have a statically known event rank. If *cat.num_classes* cannot be inferred at graph creation time, or the constant value of *cat.num_classes* is not equal to *len(components)*, or all *components* and *cat* do not have matching static batch shapes, or all components do not have matching static event shapes.

`inferpy.models.random_variable.`**`MixtureSameFamily`**(**args*, ***kwargs*)

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for MixtureSameFamily.

See MixtureSameFamily for more details.

Returns RandomVariable.

Original Docstring for Distribution

Construct a *MixtureSameFamily* distribution.

Parameters

- **mixture_distribution** – *tfp.distributions.Categorical*-like instance. Manages the probability of selecting components. The number of categories must match the rightmost batch dimension of the *components_distribution*. Must have either scalar *batch_shape* or *batch_shape* matching *components_distribution.batch_shape[:-1]*.
- **components_distribution** – *tfp.distributions.Distribution*-like instance. Right-most batch dimension indexes components.
- **reparameterize** – Python *bool*, default *False*. Whether to reparameterize samples of the distribution using implicit reparameterization gradients [(Figurnov et al., 2018)][1]. The gradients for the mixture logits are equivalent to the ones described by [(Graves, 2016)][2].

The gradients for the components parameters are also computed using implicit reparameterization (as opposed to ancestral sampling), meaning that all components are updated every step. Only works when:

(1) components_distribution is fully reparameterized;

(2) components_distribution is either a scalar distribution or fully factorized (tfd.Independent applied to a scalar distribution); (3) batch shape has a known rank.

Experimental, may be slow and produce infs/NaNs.

- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined.
- **name** – Python *str* name prefixed to Ops created by this class.

Raises

- `ValueError` – *if not dtype_util.is_integer(mixture_distribution.dtype)*.
- `ValueError` – if mixture_distribution does not have scalar *event_shape*.
- `ValueError` – if *mixture_distribution.batch_shape* and *components_distribution.batch_shape[:-1]* are both fully defined and the former is neither scalar nor equal to the latter.
- `ValueError` – if *mixture_distribution* categories does not equal *components_distribution* rightmost batch shape.

References

[1]: Michael Figurnov, Shakir Mohamed and Andriy Mnih. Implicit reparameterization gradients. In _Neural Information Processing Systems_, 2018. https://arxiv.org/abs/1805.08498

[2]: Alex Graves. Stochastic Backpropagation through Mixture Density Distributions. _arXiv_, 2016. https://arxiv.org/abs/1607.05690

`inferpy.models.random_variable.`**`Multinomial`**`(*args, **kwargs)`

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for Multinomial.

See Multinomial for more details.

Returns RandomVariable.

Original Docstring for Distribution

Initialize a batch of Multinomial distributions.

Parameters

- **total_count** – Non-negative floating point tensor with shape broadcastable to *[N1,..., Nm]* with *m >= 0*. Defines this as a batch of *N1 x ... x Nm* different Multinomial distributions. Its components should be equal to integer values.
- **logits** – Floating point tensor representing unnormalized log-probabilities of a positive event with shape broadcastable to *[N1,..., Nm, K] m >= 0*, and the same dtype as *total_count*. Defines this as a batch of *N1 x ... x Nm* different *K* class Multinomial distributions. Only one of *logits* or *probs* should be passed in.
- **probs** – Positive floating point tensor with shape broadcastable to *[N1,..., Nm, K] m >= 0* and same dtype as *total_count*. Defines this as a batch of *N1 x ... x Nm* different *K* class Multinomial distributions. *probs*'s components in the last portion of its shape should sum to *1*. Only one of *logits* or *probs* should be passed in.
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined.
- **name** – Python *str* name prefixed to Ops created by this class.

inferpy.models.random_variable.**MultivariateNormalDiag**(**args*, ***kwargs*)

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for MultivariateNormalDiag.

See MultivariateNormalDiag for more details.

Returns RandomVariable.

Original Docstring for Distribution

Construct Multivariate Normal distribution on *R^k*.

The *batch_shape* is the broadcast shape between *loc* and *scale* arguments.

The *event_shape* is given by last dimension of the matrix implied by *scale*. The last dimension of *loc* (if provided) must broadcast with this.

Recall that *covariance = scale @ scale.T*. A (non-batch) *scale* matrix is:

```
`none scale = diag(scale_diag + scale_identity_multiplier * ones(k)) `
```

where:

- *scale_diag.shape = [k]*, and,
- *scale_identity_multiplier.shape = []*.

Additional leading dimensions (if any) will index batches.

If both *scale_diag* and *scale_identity_multiplier* are *None*, then *scale* is the Identity matrix.

Parameters

- **loc** – Floating-point *Tensor*. If this is set to *None*, *loc* is implicitly *0*. When specified, may have shape *[B1, ..., Bb, k]* where *b >= 0* and *k* is the event size.
- **scale_diag** – Non-zero, floating-point *Tensor* representing a diagonal matrix added to *scale*. May have shape *[B1, ..., Bb, k]*, *b >= 0*, and characterizes *b*-batches of *k x k* diagonal matrices added to *scale*. When both *scale_identity_multiplier* and *scale_diag* are *None* then *scale* is the *Identity*.
- **scale_identity_multiplier** – Non-zero, floating-point *Tensor* representing a scaled-identity-matrix added to *scale*. May have shape *[B1, ..., Bb]*, *b >= 0*, and characterizes *b*-batches of scaled *k x k* identity matrices added to *scale*. When both *scale_identity_multiplier* and *scale_diag* are *None* then *scale* is the *Identity*.
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined.
- **name** – Python *str* name prefixed to Ops created by this class.

Raises `ValueError` – if at most *scale_identity_multiplier* is specified.

`inferpy.models.random_variable.`**`MultivariateNormalDiagPlusLowRank`**(**args*, ***kwargs*)

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for MultivariateNormalDiagPlusLowRank.

See MultivariateNormalDiagPlusLowRank for more details.

Returns RandomVariable.

Original Docstring for Distribution

Construct Multivariate Normal distribution on *R^k*.

The *batch_shape* is the broadcast shape between *loc* and *scale* arguments.

The *event_shape* is given by last dimension of the matrix implied by *scale*. The last dimension of *loc* (if provided) must broadcast with this.

Recall that *covariance = scale @ scale.T*. A (non-batch) *scale* matrix is:

"""none scale = diag(scale_diag + scale_identity_multiplier ones(k)) +

scale_perturb_factor @ diag(scale_perturb_diag) @ scale_perturb_factor.T

"""

where:

- *scale_diag.shape = [k]*,

- *scale_identity_multiplier.shape = []*,
- *scale_perturb_factor.shape = [k, r]*, typically *k >> r*, and,
- *scale_perturb_diag.shape = [r]*.

Additional leading dimensions (if any) will index batches.

If both *scale_diag* and *scale_identity_multiplier* are *None*, then *scale* is the Identity matrix.

Parameters

- **loc** – Floating-point *Tensor*. If this is set to *None*, *loc* is implicitly *0*. When specified, may have shape *[B1, ..., Bb, k]* where *b >= 0* and *k* is the event size.
- **scale_diag** – Non-zero, floating-point *Tensor* representing a diagonal matrix added to *scale*. May have shape *[B1, ..., Bb, k]*, *b >= 0*, and characterizes *b*-batches of *k x k* diagonal matrices added to *scale*. When both *scale_identity_multiplier* and *scale_diag* are *None* then *scale* is the *Identity*.
- **scale_identity_multiplier** – Non-zero, floating-point *Tensor* representing a scaled-identity-matrix added to *scale*. May have shape *[B1, ..., Bb]*, *b >= 0*, and characterizes *b*-batches of scaled *k x k* identity matrices added to *scale*. When both *scale_identity_multiplier* and *scale_diag* are *None* then *scale* is the *Identity*.
- **scale_perturb_factor** – Floating-point *Tensor* representing a rank-*r* perturbation added to *scale*. May have shape *[B1, ..., Bb, k, r]*, *b >= 0*, and characterizes *b*-batches of rank-*r* updates to *scale*. When *None*, no rank-*r* update is added to *scale*.
- **scale_perturb_diag** – Floating-point *Tensor* representing a diagonal matrix inside the rank-*r* perturbation added to *scale*. May have shape *[B1, ..., Bb, r]*, *b >= 0*, and characterizes *b*-batches of *r x r* diagonal matrices inside the perturbation added to *scale*. When *None*, an identity matrix is used inside the perturbation. Can only be specified if *scale_perturb_factor* is also specified.
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined.
- **name** – Python *str* name prefixed to Ops created by this class.

Raises `ValueError` – if at most *scale_identity_multiplier* is specified.

`inferpy.models.random_variable.`**`MultivariateNormalDiagWithSoftplusScale`**`(`*`*args`*, *`**kwargs`*`)`

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for MultivariateNormalDiagWithSoftplusScale.

See MultivariateNormalDiagWithSoftplusScale for more details.

Returns RandomVariable.

Original Docstring for Distribution

DEPRECATED FUNCTION

Warning: THIS FUNCTION IS DEPRECATED. It will be removed after 2019-06-05. Instructions for updating: MultivariateNormalDiagWithSoftplusScale is deprecated, use MultivariateNormalDiag(loc=loc, scale_diag=tf.nn.softplus(scale_diag)) instead.

inferpy.models.random_variable.**MultivariateNormalFullCovariance**(**args*, ***kwargs*)

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for MultivariateNormalFullCovariance.

See MultivariateNormalFullCovariance for more details.

Returns RandomVariable.

Original Docstring for Distribution

Construct Multivariate Normal distribution on *R^k*.

The *batch_shape* is the broadcast shape between *loc* and *covariance_matrix* arguments.

The *event_shape* is given by last dimension of the matrix implied by *covariance_matrix*. The last dimension of *loc* (if provided) must broadcast with this.

A non-batch *covariance_matrix* matrix is a *k x k* symmetric positive definite matrix. In other words it is (real) symmetric with all eigenvalues strictly positive.

Additional leading dimensions (if any) will index batches.

Parameters

- **loc** – Floating-point *Tensor*. If this is set to *None*, *loc* is implicitly *0*. When specified, may have shape *[B1, ..., Bb, k]* where *b >= 0* and *k* is the event size.
- **covariance_matrix** – Floating-point, symmetric positive definite *Tensor* of same *dtype* as *loc*. The strict upper triangle of *covariance_matrix* is ignored, so if *covariance_matrix* is not symmetric no error will be raised (unless *validate_args is True*). *covariance_matrix* has shape *[B1, ..., Bb, k, k]* where *b >= 0* and *k* is the event size.
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined.
- **name** – Python *str* name prefixed to Ops created by this class.

Raises `ValueError` – if neither *loc* nor *covariance_matrix* are specified.

inferpy.models.random_variable.**MultivariateNormalLinearOperator**(**args*, ***kwargs*)

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for MultivariateNormalLinearOperator.

See MultivariateNormalLinearOperator for more details.

Returns RandomVariable.

Original Docstring for Distribution

Construct Multivariate Normal distribution on *R^k*.

The *batch_shape* is the broadcast shape between *loc* and *scale* arguments.

The *event_shape* is given by last dimension of the matrix implied by *scale*. The last dimension of *loc* (if provided) must broadcast with this.

Recall that *covariance = scale @ scale.T*.

Additional leading dimensions (if any) will index batches.

Parameters

- **loc** – Floating-point *Tensor*. If this is set to *None*, *loc* is implicitly *0.* When specified, may have shape *[B1, ..., Bb, k]* where *b >= 0* and *k* is the event size.
- **scale** – Instance of *LinearOperator* with same *dtype* as *loc* and shape *[B1, ..., Bb, k, k].*
- **validate_args** – Python *bool*, default *False*. Whether to validate input with asserts. If *validate_args* is *False*, and the inputs are invalid, correct behavior is not guaranteed.
- **allow_nan_stats** – Python *bool*, default *True*. If *False*, raise an exception if a statistic (e.g. mean/mode/etc...) is undefined for any batch member If *True*, batch members with valid parameters leading to undefined statistics will return NaN for this statistic.
- **name** – The name to give Ops created by the initializer.

Raises

- ValueError – if *scale* is unspecified.
- TypeError – if not *scale.dtype.is_floating*

inferpy.models.random_variable.**MultivariateNormalTriL**(**args*, ***kwargs*)

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for MultivariateNormalTriL.

See MultivariateNormalTriL for more details.

Returns RandomVariable.

Original Docstring for Distribution

Construct Multivariate Normal distribution on *R^k*.

The *batch_shape* is the broadcast shape between *loc* and *scale* arguments.

The *event_shape* is given by last dimension of the matrix implied by *scale*. The last dimension of *loc* (if provided) must broadcast with this.

Recall that *covariance* = *scale @ scale.T*. A (non-batch) *scale* matrix is:

`none scale = scale_tril `

where *scale_tril* is lower-triangular *k x k* matrix with non-zero diagonal, i.e., *tf.diag_part(scale_tril) != 0*.

Additional leading dimensions (if any) will index batches.

Parameters

- **loc** – Floating-point *Tensor*. If this is set to *None*, *loc* is implicitly *0*. When specified, may have shape *[B1, ..., Bb, k]* where *b >= 0* and *k* is the event size.
- **scale_tril** – Floating-point, lower-triangular *Tensor* with non-zero diagonal elements. *scale_tril* has shape *[B1, ..., Bb, k, k]* where *b >= 0* and *k* is the event size.
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined.
- **name** – Python *str* name prefixed to Ops created by this class.

Raises `ValueError` – if neither *loc* nor *scale_tril* are specified.

inferpy.models.random_variable.**MultivariateStudentTLinearOperator**(**args*, ***kwargs*)

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for MultivariateStudentTLinearOperator.

See MultivariateStudentTLinearOperator for more details.

Returns RandomVariable.

Original Docstring for Distribution

Construct Multivariate Student's t-distribution on *R^k*.

The *batch_shape* is the broadcast shape between *df*, *loc* and *scale* arguments.

The *event_shape* is given by last dimension of the matrix implied by *scale*. The last dimension of *loc* must broadcast with this.

Additional leading dimensions (if any) will index batches.

Parameters

- **df** – A positive floating-point *Tensor*. Has shape *[B1, ..., Bb]* where *b >= 0*.
- **loc** – Floating-point *Tensor*. Has shape *[B1, ..., Bb, k]* where *k* is the event size.
- **scale** – Instance of *LinearOperator* with a floating *dtype* and shape *[B1, ..., Bb, k, k]*.
- **validate_args** – Python *bool*, default *False*. Whether to validate input with asserts. If *validate_args* is *False*, and the inputs are invalid, correct behavior is not guaranteed.
- **allow_nan_stats** – Python *bool*, default *True*. If *False*, raise an exception if a statistic (e.g. mean/variance/etc...) is undefined for any batch member If *True*, batch members with valid parameters leading to undefined statistics will return NaN for this statistic.
- **name** – The name to give Ops created by the initializer.

Raises

- `TypeError` – if not *scale.dtype.is_floating*.
- `ValueError` – if not *scale.is_positive_definite*.

`inferpy.models.random_variable.`**`NegativeBinomial`**`(*args, **kwargs)`

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for NegativeBinomial.

See NegativeBinomial for more details.

Returns RandomVariable.

Original Docstring for Distribution

Construct NegativeBinomial distributions.

Parameters

- **total_count** – Non-negative floating-point *Tensor* with shape broadcastable to *[B1,..., Bb]* with *b >= 0* and the same dtype as *probs* or *logits*. Defines this as a batch of *N1 x ... x Nm* different Negative Binomial distributions. In practice, this represents the number of negative Bernoulli trials to stop at (the *total_count* of failures), but this is still a valid distribution when *total_count* is a non-integer.
- **logits** – Floating-point *Tensor* with shape broadcastable to *[B1, ..., Bb]* where *b >= 0* indicates the number of batch dimensions. Each entry represents logits for the probability of success for independent Negative Binomial distributions and must be in the open interval *(-inf, inf)*. Only one of *logits* or *probs* should be specified.
- **probs** – Positive floating-point *Tensor* with shape broadcastable to *[B1, ..., Bb]* where *b >= 0* indicates the number of batch dimensions. Each entry represents the probability of

success for independent Negative Binomial distributions and must be in the open interval *(0, 1)*. Only one of *logits* or *probs* should be specified.

- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined.
- **name** – Python *str* name prefixed to Ops created by this class.

inferpy.models.random_variable.**Normal**(**args*, ***kwargs*)

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for Normal.

See Normal for more details.

Returns RandomVariable.

Original Docstring for Distribution

Construct Normal distributions with mean and stddev *loc* and *scale*.

The parameters *loc* and *scale* must be shaped in a way that supports broadcasting (e.g. *loc* + *scale* is a valid operation).

Parameters

- **loc** – Floating point tensor; the means of the distribution(s).
- **scale** – Floating point tensor; the stddevs of the distribution(s). Must contain only positive values.
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined.
- **name** – Python *str* name prefixed to Ops created by this class.

Raises TypeError – if *loc* and *scale* have different *dtype*.

inferpy.models.random_variable.**OneHotCategorical**(**args*, ***kwargs*)

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.

- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for OneHotCategorical.

See OneHotCategorical for more details.

Returns RandomVariable.

Original Docstring for Distribution

Initialize OneHotCategorical distributions using class log-probabilities.

Parameters

- **logits** – An N-D *Tensor*, *N >= 1*, representing the log probabilities of a set of Categorical distributions. The first *N - 1* dimensions index into a batch of independent distributions and the last dimension represents a vector of logits for each class. Only one of *logits* or *probs* should be passed in.
- **probs** – An N-D *Tensor*, *N >= 1*, representing the probabilities of a set of Categorical distributions. The first *N - 1* dimensions index into a batch of independent distributions and the last dimension represents a vector of probabilities for each class. Only one of *logits* or *probs* should be passed in.
- **dtype** – The type of the event samples (default: int32).
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined.
- **name** – Python *str* name prefixed to Ops created by this class.

inferpy.models.random_variable.**Pareto**(**args*, ***kwargs*)

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for Pareto.

See Pareto for more details.

Returns RandomVariable.

Original Docstring for Distribution

Construct Pareto distribution with *concentration* and *scale*.

Parameters

- **concentration** – Floating point tensor. Must contain only positive values.
- **scale** – Floating point tensor, equivalent to *mode*. *scale* also restricts the domain of this distribution to be in *[scale, inf)*. Must contain only positive values. Default value: *1*.

- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs. Default value: *False* (i.e. do not validate args).
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined. Default value: *True*.
- **name** – Python *str* name prefixed to Ops created by this class. Default value: 'Pareto'.

inferpy.models.random_variable.**Poisson**(**args*, ***kwargs*)

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for Poisson.

See Poisson for more details.

Returns RandomVariable.

Original Docstring for Distribution

Initialize a batch of Poisson distributions.

Parameters

- **rate** – Floating point tensor, the rate parameter. *rate* must be positive. Must specify exactly one of *rate* and *log_rate*.
- **log_rate** – Floating point tensor, the log of the rate parameter. Must specify exactly one of *rate* and *log_rate*.
- **interpolate_nondiscrete** – Python *bool*. When *False*, *log_prob* returns *-inf* (and *prob* returns *0*) for non-integer inputs. When *True*, *log_prob* evaluates the continuous function *k * log_rate - lgamma(k+1) - rate*, which matches the Poisson pmf at integer arguments *k* (note that this function is not itself a normalized probability log-density). Default value: *True*.
- **validate_args** – Python *bool*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs. Default value: *False*.
- **allow_nan_stats** – Python *bool*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined. Default value: *True*.
- **name** – Python *str* name prefixed to Ops created by this class.

Raises

- ValueError – if none or both of *rate*, *log_rate* are specified.
- TypeError – if *rate* is not a float-type.
- TypeError – if *log_rate* is not a float-type.

inferpy.models.random_variable.**PoissonLogNormalQuadratureCompound**(**args*, ***kwargs*)

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for PoissonLogNormalQuadratureCompound.

See PoissonLogNormalQuadratureCompound for more details.

Returns RandomVariable.

Original Docstring for Distribution

Constructs the PoissonLogNormalQuadratureCompound‘.

Note: *probs* returned by (optional) *quadrature_fn* are presumed to be either a length-*quadrature_size* vector or a batch of vectors in 1-to-1 correspondence with the returned *grid*. (I.e., broadcasting is only partially supported.)

Parameters

- **loc** – *float*-like (batch of) scalar *Tensor*; the location parameter of the LogNormal prior.
- **scale** – *float*-like (batch of) scalar *Tensor*; the scale parameter of the LogNormal prior.
- **quadrature_size** – Python *int* scalar representing the number of quadrature points.
- **quadrature_fn** – Python callable taking *loc*, *scale*, *quadrature_size*, *validate_args* and returning *tuple(grid, probs)* representing the LogNormal grid and corresponding normalized weight. normalized) weight. Default value: *quadrature_scheme_lognormal_quantiles*.
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value “*NaN*” to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic’s batch members are undefined.
- **name** – Python *str* name prefixed to Ops created by this class.

Raises TypeError – if *quadrature_grid* and *quadrature_probs* have different base *dtype*.

inferpy.models.random_variable.**QuantizedDistribution**(**args*, ***kwargs*)

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for QuantizedDistribution.

See QuantizedDistribution for more details.

Returns RandomVariable.

Original Docstring for Distribution

Construct a Quantized Distribution representing *Y = ceiling(X)*.

Some properties are inherited from the distribution defining *X*. Example: *allow_nan_stats* is determined for this *QuantizedDistribution* by reading the *distribution*.

Parameters

- **distribution** – The base distribution class to transform. Typically an instance of *Distribution*.
- **low** – *Tensor* with same *dtype* as this distribution and shape able to be added to samples. Should be a whole number. Default *None*. If provided, base distribution's *prob* should be defined at *low*.
- **high** – *Tensor* with same *dtype* as this distribution and shape able to be added to samples. Should be a whole number. Default *None*. If provided, base distribution's *prob* should be defined at *high - 1*. *high* must be strictly greater than *low*.
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **name** – Python *str* name prefixed to Ops created by this class.

Raises

- `TypeError` – If *dist_cls* is not a subclass of *Distribution* or continuous.
- `NotImplementedError` – If the base distribution does not implement *cdf*.

class inferpy.models.random_variable.**RandomVariable**(*var*, *name*, *is_datamodel*, *ed_cls*, *var_args*, *var_kwargs*, *sample_shape*, *is_observed*, *observed_value*)

Bases: `object`

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

build_in_session(*sess*)

Allow to build a copy of the random variable but running previously each parameter in the tf session. This way, it uses the value of each tf variable or placeholder as a tensor, not as a tf variable or placeholder. If this random variable is a ed random variable directly assigned to .var, we cannot re-create it. In this case, return self. :param sess: tf session used to run each parameter used to build this random variable. :returns: the random variable object

copy()

Makes a of the current random variable where the distribution parameters are fixed. :return: new object of class RandomVariable

property type

inferpy.models.random_variable.**RelaxedBernoulli**(**args*, ***kwargs*)

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for RelaxedBernoulli.

See RelaxedBernoulli for more details.

Returns RandomVariable.

Original Docstring for Distribution

Construct RelaxedBernoulli distributions.

Parameters

- **temperature** – An 0-D *Tensor*, representing the temperature of a set of RelaxedBernoulli distributions. The temperature should be positive.
- **logits** – An N-D *Tensor* representing the log-odds of a positive event. Each entry in the *Tensor* parametrizes an independent RelaxedBernoulli distribution where the probability of an event is sigmoid(logits). Only one of *logits* or *probs* should be passed in.
- **probs** – An N-D *Tensor* representing the probability of a positive event. Each entry in the *Tensor* parameterizes an independent Bernoulli distribution. Only one of *logits* or *probs* should be passed in.
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined.
- **name** – Python *str* name prefixed to Ops created by this class.

Raises `ValueError` – If both *probs* and *logits* are passed, or if neither.

`inferpy.models.random_variable.`**`RelaxedOneHotCategorical`**`(*args, **kwargs)`

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for RelaxedOneHotCategorical.

See RelaxedOneHotCategorical for more details.

Returns RandomVariable.

Original Docstring for Distribution

Initialize RelaxedOneHotCategorical using class log-probabilities.

Parameters

- **temperature** – An 0-D *Tensor*, representing the temperature of a set of RelaxedOneHotCategorical distributions. The temperature should be positive.
- **logits** – An N-D *Tensor*, *N >= 1*, representing the log probabilities of a set of RelaxedOneHotCategorical distributions. The first *N - 1* dimensions index into a batch of independent distributions and the last dimension represents a vector of logits for each class. Only one of *logits* or *probs* should be passed in.
- **probs** – An N-D *Tensor*, *N >= 1*, representing the probabilities of a set of RelaxedOneHotCategorical distributions. The first *N - 1* dimensions index into a batch of independent distributions and the last dimension represents a vector of probabilities for each class. Only one of *logits* or *probs* should be passed in.
- **validate_args** – Unused in this distribution.
- **allow_nan_stats** – Python *bool*, default *True*. If *False*, raise an exception if a statistic (e.g. mean/mode/etc...) is undefined for any batch member. If *True*, batch members with valid parameters leading to undefined statistics will return NaN for this statistic.
- **name** – A name for this distribution (optional).

inferpy.models.random_variable.**Sample**(**args*, ***kwargs*)

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for Sample.

See Sample for more details.

Returns RandomVariable.

Original Docstring for Distribution

Construct the *Sample* distribution.

Parameters

- **distribution** – The base distribution instance to transform. Typically an instance of *Distribution*.
- **sample_shape** – *int* scalar or vector *Tensor* representing the shape of a single sample.
- **validate_args** – Python *bool*. Whether to validate input with asserts. If *validate_args* is *False*, and the inputs are invalid, correct behavior is not guaranteed.
- **name** – The name for ops managed by the distribution. Default value: *None* (i.e., *'Sample' + distribution.name*).

inferpy.models.random_variable.**SinhArcsinh**(**args*, ***kwargs*)

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.

- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for SinhArcsinh.

See SinhArcsinh for more details.

Returns RandomVariable.

Original Docstring for Distribution

Construct SinhArcsinh distribution on *(-inf, inf)*.

Arguments *(loc, scale, skewness, tailweight)* must have broadcastable shape (indexing batch dimensions). They must all have the same *dtype*.

Parameters

- **loc** – Floating-point *Tensor*.
- **scale** – *Tensor* of same *dtype* as *loc*.
- **skewness** – Skewness parameter. Default is *0.0* (no skew).
- **tailweight** – Tailweight parameter. Default is *1.0* (unchanged tailweight)
- **distribution** – *tf.Distribution*-like instance. Distribution that is transformed to produce this distribution. Default is *tfd.Normal(0., 1.)*. Must be a scalar-batch, scalar-event distribution. Typically *distribution.reparameterization_type* = *FULLY_REPARAMETERIZED* or it is a function of non-trainable parameters. WARNING: If you backprop through a *SinhArcsinh* sample and *distribution* is not *FULLY_REPARAMETERIZED* yet is a function of trainable variables, then the gradient will be incorrect!
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined.
- **name** – Python *str* name prefixed to Ops created by this class.

inferpy.models.random_variable.**StudentT**(**args*, ***kwargs*)

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for StudentT.

See StudentT for more details.

Returns RandomVariable.

Original Docstring for Distribution

Construct Student's t distributions.

The distributions have degree of freedom *df*, mean *loc*, and scale *scale*.

The parameters *df*, *loc*, and *scale* must be shaped in a way that supports broadcasting (e.g. *df* + *loc* + *scale* is a valid operation).

Parameters

- **df** – Floating-point *Tensor*. The degrees of freedom of the distribution(s). *df* must contain only positive values.
- **loc** – Floating-point *Tensor*. The mean(s) of the distribution(s).
- **scale** – Floating-point *Tensor*. The scaling factor(s) for the distribution(s). Note that *scale* is not technically the standard deviation of this distribution but has semantics more similar to standard deviation than variance.
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined.
- **name** – Python *str* name prefixed to Ops created by this class.

Raises `TypeError` – if loc and scale are different dtypes.

`inferpy.models.random_variable.`**`StudentTProcess`**`(`*args*, ***kwargs*`)`

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for StudentTProcess.

See StudentTProcess for more details.

Returns RandomVariable.

Original Docstring for Distribution

Instantiate a StudentTProcess Distribution.

Parameters

- **df** – Positive Floating-point *Tensor* representing the degrees of freedom. Must be greater than 2.
- **kernel** – *PositiveSemidefiniteKernel*-like instance representing the TP's covariance function.
- **index_points** – *float Tensor* representing finite (batch of) vector(s) of points in the index set over which the TP is defined. Shape has the form *[b1, . . . , bB, e, f1, . . . , fF]* where *F* is the number of feature dimensions and must equal *kernel.feature_ndims* and *e* is the number (size) of index points in each batch. Ultimately this distribution corresponds to a *e*-dimensional multivariate Student's T. The batch shape must be broadcastable with *kernel.batch_shape* and any batch dims yielded by *mean_fn*.

- **mean_fn** – Python *callable* that acts on *index_points* to produce a (batch of) vector(s) of mean values at *index_points*. Takes a *Tensor* of shape *[b1, ..., bB, f1, ..., fF]* and returns a *Tensor* whose shape is broadcastable with *[b1, ..., bB]*. Default value: *None* implies constant zero function.
- **jitter** – *float* scalar *Tensor* added to the diagonal of the covariance matrix to ensure positive definiteness of the covariance matrix. Default value: *1e-6*.
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs. Default value: *False*.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined. Default value: *False*.
- **name** – Python *str* name prefixed to Ops created by this class. Default value: "StudentTProcess".

Raises `ValueError` – if *mean_fn* is not *None* and is not callable.

`inferpy.models.random_variable.`**`TransformedDistribution`**(**args*, ***kwargs*)

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for TransformedDistribution.

See TransformedDistribution for more details.

Returns RandomVariable.

Original Docstring for Distribution

Construct a Transformed Distribution.

Parameters

- **distribution** – The base distribution instance to transform. Typically an instance of *Distribution*.
- **bijector** – The object responsible for calculating the transformation. Typically an instance of *Bijector*.
- **batch_shape** – *integer* vector *Tensor* which overrides *distribution batch_shape*; valid only if *distribution.is_scalar_batch()*.
- **event_shape** – *integer* vector *Tensor* which overrides *distribution event_shape*; valid only if *distribution.is_scalar_event()*.
- **kwargs_split_fn** – Python *callable* which takes a kwargs *dict* and returns a tuple of kwargs *dict's for each of the 'distribution* and *bijector* parameters respectively. Default value: *_default_kwargs_split_fn* (i.e.,

  'lambda kwargs: (kwargs.get('distribution_kwargs', {}),
  kwargs.get('bijector_kwargs', {}))')

- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **parameters** – Locals dict captured by subclass constructor, to be used for copy/slice re-instantiation operations.
- **name** – Python *str* name prefixed to Ops created by this class. Default: *bijector.name* + *distribution.name*.

inferpy.models.random_variable.**Triangular**(**args*, ***kwargs*)

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for Triangular.

See Triangular for more details.

Returns RandomVariable.

Original Docstring for Distribution

Initialize a batch of Triangular distributions.

Parameters

- **low** – Floating point tensor, lower boundary of the output interval. Must have *low* < *high*. Default value: *0*.
- **high** – Floating point tensor, upper boundary of the output interval. Must have *low* < *high*. Default value: *1*.
- **peak** – Floating point tensor, mode of the output interval. Must have *low* <= *peak* and *peak* <= *high*. Default value: *0.5*.
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs. Default value: *False*.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined. Default value: *True*.
- **name** – Python *str* name prefixed to Ops created by this class. Default value: *'Triangular'*.

Raises InvalidArgumentError – if *validate_args*=*True* and one of the following is True: * *low* >= *high*. * *peak* > *high*. * *low* > *peak*.

inferpy.models.random_variable.**TruncatedNormal**(**args*, ***kwargs*)

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.

- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for TruncatedNormal.

See TruncatedNormal for more details.

Returns RandomVariable.

Original Docstring for Distribution

Construct TruncatedNormal.

All parameters of the distribution will be broadcast to the same shape, so the resulting distribution will have a batch_shape of the broadcast shape of all parameters.

Parameters

- **loc** – Floating point tensor; the mean of the normal distribution(s) (note that the mean of the resulting distribution will be different since it is modified by the bounds).
- **scale** – Floating point tensor; the std deviation of the normal distribution(s).
- **low** – *float Tensor* representing lower bound of the distribution's support. Must be such that *low* < *high*.
- **high** – *float Tensor* representing upper bound of the distribution's support. Must be such that *low* < *high*.
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked at run-time.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined.
- **name** – Python *str* name prefixed to Ops created by this class.

inferpy.models.random_variable.**Uniform**(**args*, ***kwargs*)

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for Uniform.

See Uniform for more details.

Returns RandomVariable.

Original Docstring for Distribution

Initialize a batch of Uniform distributions.

Parameters

- **low** – Floating point tensor, lower boundary of the output interval. Must have *low* < *high*.
- **high** – Floating point tensor, upper boundary of the output interval. Must have *low* < *high*.

- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined.
- **name** – Python *str* name prefixed to Ops created by this class.

Raises `InvalidArgumentError` – if *low* >= *high* and *validate_args=False*.

`inferpy.models.random_variable.`**`VariationalGaussianProcess`**`(`**args*, ***kwargs*`)`

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for VariationalGaussianProcess.

See VariationalGaussianProcess for more details.

Returns RandomVariable.

Original Docstring for Distribution

Instantiate a VariationalGaussianProcess Distribution.

Parameters

- **kernel** – *PositiveSemidefiniteKernel*-like instance representing the GP's covariance function.
- **index_points** – *float Tensor* representing finite (batch of) vector(s) of points in the index set over which the VGP is defined. Shape has the form *[b1, ..., bB, e1, f1, ..., fF]* where *F* is the number of feature dimensions and must equal *kernel.feature_ndims* and *e1* is the number (size) of index points in each batch (we denote it *e1* to distinguish it from the numer of inducing index points, denoted *e2* below). Ultimately the VariationalGaussianProcess distribution corresponds to an *e1*-dimensional multivariate normal. The batch shape must be broadcastable with *kernel.batch_shape*, the batch shape of *inducing_index_points*, and any batch dims yielded by *mean_fn*.
- **inducing_index_points** – *float Tensor* of locations of inducing points in the index set. Shape has the form *[b1, ..., bB, e2, f1, ..., fF]*, just like *index_points*. The batch shape components needn't be identical to those of *index_points*, but must be broadcast compatible with them.
- **variational_inducing_observations_loc** – *float Tensor*; the mean of the (full-rank Gaussian) variational posterior over function values at the inducing points, conditional on observed data. Shape has the form *[b1, ..., bB, e2]*, where *b1, ..., bB* is broadcast compatible with other parameters' batch shapes, and *e2* is the number of inducing points.
- **variational_inducing_observations_scale** – *float Tensor*; the scale matrix of the (full-rank Gaussian) variational posterior over function values at the inducing points, conditional on observed data. Shape has the form *[b1, ..., bB, e2, e2]*, where *b1, ..., bB* is broadcast compatible with other parameters and *e2* is the number of inducing points.

- **mean_fn** – Python *callable* that acts on index points to produce a (batch of) vector(s) of mean values at those index points. Takes a *Tensor* of shape *[b1, ..., bB, f1, ..., fF]* and returns a *Tensor* whose shape is (broadcastable with) *[b1, ..., bB]*. Default value: *None* implies constant zero function.
- **observation_noise_variance** – *float Tensor* representing the variance of the noise in the Normal likelihood distribution of the model. May be batched, in which case the batch shape must be broadcastable with the shapes of all other batched parameters (*kernel.batch_shape*, *index_points*, etc.). Default value: *0.*
- **predictive_noise_variance** – *float Tensor* representing additional variance in the posterior predictive model. If *None*, we simply re-use *observation_noise_variance* for the posterior predictive noise. If set explicitly, however, we use the given value. This allows us, for example, to omit predictive noise variance (by setting this to zero) to obtain noiseless posterior predictions of function values, conditioned on noisy observations.
- **jitter** – *float* scalar *Tensor* added to the diagonal of the covariance matrix to ensure positive definiteness of the covariance matrix. Default value: *1e-6*.
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs. Default value: *False*.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined. Default value: *False*.
- **name** – Python *str* name prefixed to Ops created by this class. Default value: "VariationalGaussianProcess".

Raises `ValueError` – if *mean_fn* is not *None* and is not callable.

`inferpy.models.random_variable.`**`VectorDeterministic`**`(*args, **kwargs)`

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for VectorDeterministic.

See VectorDeterministic for more details.

Returns RandomVariable.

Original Docstring for Distribution

Initialize a *VectorDeterministic* distribution on *R^k*, for *k >= 0*.

Note that there is only one point in *R^0*, the "point" *[]*. So if *k = 0* then *self.prob([]) == 1*.

The *atol* and *rtol* parameters allow for some slack in *pmf* computations, e.g. due to floating-point error.

``` pmf(x; loc)

= 1, if All[Abs(x - loc) <= atol + rtol * Abs(loc)], = 0, otherwise

```

Parameters

- **loc** – Numeric *Tensor* of shape *[B1, ..., Bb, k]*, with *b >= 0*, *k >= 0* The point (or batch of points) on which this distribution is supported.
- **atol** – Non-negative *Tensor* of same *dtype* as *loc* and broadcastable shape. The absolute tolerance for comparing closeness to *loc*. Default is *0*.
- **rtol** – Non-negative *Tensor* of same *dtype* as *loc* and broadcastable shape. The relative tolerance for comparing closeness to *loc*. Default is *0*.
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined.
- **name** – Python *str* name prefixed to Ops created by this class.

inferpy.models.random_variable.**VectorDiffeomixture**(**args*, ***kwargs*)

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for VectorDiffeomixture.

See VectorDiffeomixture for more details.

Returns RandomVariable.

Original Docstring for Distribution

Constructs the VectorDiffeomixture on *R^d*.

The vector diffeomixture (VDM) approximates the compound distribution

```
`none p(x) = int p(x | z) p(z) dz, where z is in the K-simplex, and p(x | z) := p(x | loc=sum_k z[k] loc[k], scale=sum_k z[k] scale[k]) `
```

Parameters

- **mix_loc** – *float*-like *Tensor* with shape *[b1, ..., bB, K-1]*. In terms of samples, larger *mix_loc[..., k] ==> Z* is more likely to put more weight on its *kth* component.
- **temperature** – *float*-like *Tensor*. Broadcastable with *mix_loc*. In terms of samples, smaller *temperature* means one component is more likely to dominate. I.e., smaller *temperature* makes the VDM look more like a standard mixture of *K* components.
- **distribution** – *tfp.distributions.Distribution*-like instance. Distribution from which *d* iid samples are used as input to the selected affine transformation. Must be a scalar-batch, scalar-event distribution. Typically *distribution.reparameterization_type = FULLY_REPARAMETERIZED* or it is a function of non-trainable parameters. WARNING: If you backprop through a VectorDiffeomixture sample and the *distribution* is not *FULLY_REPARAMETERIZED* yet is a function of trainable variables, then the gradient will be incorrect!

- **loc** – Length-*K* list of *float*-type *Tensor's. The 'k*-th element represents the *shift* used for the *k*-th affine transformation. If the *k*-th item is *None*, *loc* is implicitly *0*. When specified, must have shape *[B1, ..., Bb, d]* where *b >= 0* and *d* is the event size.
- **scale** – Length-*K* list of *LinearOperator's. Each should be positive-definite and operate on a 'd*-dimensional vector space. The *k*-th element represents the *scale* used for the *k*-th affine transformation. *LinearOperator's must have shape '[B1, ..., Bb, d, d]*, *b >= 0*, i.e., characterizes *b*-batches of *d x d* matrices
- **quadrature_size** – Python *int* scalar representing number of quadrature points. Larger *quadrature_size* means *q_N(x)* better approximates *p(x)*.
- **quadrature_fn** – Python callable taking *normal_loc*, *normal_scale*, *quadrature_size*, *validate_args* and returning *tuple(grid, probs)* representing the SoftmaxNormal grid and corresponding normalized weight. normalized) weight. Default value: *quadrature_scheme_softmaxnormal_quantiles*.
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined.
- **name** – Python *str* name prefixed to Ops created by this class.

Raises

- `ValueError` – if *not scale or len(scale) < 2*.
- `ValueError` – if *len(loc) != len(scale)*
- `ValueError` – if *quadrature_grid_and_probs is not None* and *len(quadrature_grid_and_probs[0]) != len(quadrature_grid_and_probs[1])*
- `ValueError` – if *validate_args* and any not scale.is_positive_definite.
- `TypeError` – if any scale.dtype != scale[0].dtype.
- `TypeError` – if any loc.dtype != scale[0].dtype.
- `NotImplementedError` – if *len(scale) != 2*.
- `ValueError` – if *not distribution.is_scalar_batch*.
- `ValueError` – if *not distribution.is_scalar_event*.

`inferpy.models.random_variable.`**`VectorExponentialDiag`**`(*args, **kwargs)`

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for VectorExponentialDiag.

See VectorExponentialDiag for more details.

Returns RandomVariable.

Original Docstring for Distribution

Construct Vector Exponential distribution supported on a subset of *R^k*.

The *batch_shape* is the broadcast shape between *loc* and *scale* arguments.

The *event_shape* is given by last dimension of the matrix implied by *scale*. The last dimension of *loc* (if provided) must broadcast with this.

Recall that *covariance = scale @ scale.T*.

```
`none scale = diag(scale_diag + scale_identity_multiplier * ones(k)) `
```

where:

- *scale_diag.shape = [k]*, and,
- *scale_identity_multiplier.shape = []*.

Additional leading dimensions (if any) will index batches.

If both *scale_diag* and *scale_identity_multiplier* are *None*, then *scale* is the Identity matrix.

Parameters

- **loc** – Floating-point *Tensor*. If this is set to *None*, *loc* is implicitly *0.* When specified, may have shape *[B1, ..., Bb, k]* where *b >= 0* and *k* is the event size.
- **scale_diag** – Non-zero, floating-point *Tensor* representing a diagonal matrix added to *scale*. May have shape *[B1, ..., Bb, k]*, *b >= 0*, and characterizes *b*-batches of *k x k* diagonal matrices added to *scale*. When both *scale_identity_multiplier* and *scale_diag* are *None* then *scale* is the *Identity*.
- **scale_identity_multiplier** – Non-zero, floating-point *Tensor* representing a scaled-identity-matrix added to *scale*. May have shape *[B1, ..., Bb]*, *b >= 0*, and characterizes *b*-batches of scaled *k x k* identity matrices added to *scale*. When both *scale_identity_multiplier* and *scale_diag* are *None* then *scale* is the *Identity*.
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined.
- **name** – Python *str* name prefixed to Ops created by this class.

Raises `ValueError` – if at most *scale_identity_multiplier* is specified.

`inferpy.models.random_variable.`**`VectorLaplaceDiag`**`(`*args*, ***kwargs*`)`

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for VectorLaplaceDiag.

See VectorLaplaceDiag for more details.

Returns RandomVariable.

Original Docstring for Distribution

Construct Vector Laplace distribution on *R^k*.

The *batch_shape* is the broadcast shape between *loc* and *scale* arguments.

The *event_shape* is given by last dimension of the matrix implied by *scale*. The last dimension of *loc* (if provided) must broadcast with this.

Recall that *covariance = 2 * scale @ scale.T*.

`none scale = diag(scale_diag + scale_identity_multiplier * ones(k)) `

where:

- *scale_diag.shape = [k]*, and,
- *scale_identity_multiplier.shape = []*.

Additional leading dimensions (if any) will index batches.

If both *scale_diag* and *scale_identity_multiplier* are *None*, then *scale* is the Identity matrix.

Parameters

- **loc** – Floating-point *Tensor*. If this is set to *None*, *loc* is implicitly *0*. When specified, may have shape *[B1, ..., Bb, k]* where *b >= 0* and *k* is the event size.
- **scale_diag** – Non-zero, floating-point *Tensor* representing a diagonal matrix added to *scale*. May have shape *[B1, ..., Bb, k]*, *b >= 0*, and characterizes *b*-batches of *k x k* diagonal matrices added to *scale*. When both *scale_identity_multiplier* and *scale_diag* are *None* then *scale* is the *Identity*.
- **scale_identity_multiplier** – Non-zero, floating-point *Tensor* representing a scaled-identity-matrix added to *scale*. May have shape *[B1, ..., Bb]*, *b >= 0*, and characterizes *b*-batches of scaled *k x k* identity matrices added to *scale*. When both *scale_identity_multiplier* and *scale_diag* are *None* then *scale* is the *Identity*.
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined.
- **name** – Python *str* name prefixed to Ops created by this class.

Raises `ValueError` – if at most *scale_identity_multiplier* is specified.

inferpy.models.random_variable.**VectorSinhArcsinhDiag**(**args*, ***kwargs*)

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for VectorSinhArcsinhDiag.

See VectorSinhArcsinhDiag for more details.

Returns RandomVariable.

Original Docstring for Distribution

Construct VectorSinhArcsinhDiag distribution on *R^k*.

The arguments *scale_diag* and *scale_identity_multiplier* combine to define the diagonal *scale* referred to in this class docstring:

```
`none scale = diag(scale_diag + scale_identity_multiplier * ones(k)) `
```

The *batch_shape* is the broadcast shape between *loc* and *scale* arguments.

The *event_shape* is given by last dimension of the matrix implied by *scale*. The last dimension of *loc* (if provided) must broadcast with this

Additional leading dimensions (if any) will index batches.

Parameters

- **loc** – Floating-point *Tensor*. If this is set to *None*, *loc* is implicitly *0.* When specified, may have shape *[B1, ..., Bb, k]* where *b >= 0* and *k* is the event size.
- **scale_diag** – Non-zero, floating-point *Tensor* representing a diagonal matrix added to *scale*. May have shape *[B1, ..., Bb, k]*, *b >= 0*, and characterizes *b*-batches of *k x k* diagonal matrices added to *scale*. When both *scale_identity_multiplier* and *scale_diag* are *None* then *scale* is the *Identity*.
- **scale_identity_multiplier** – Non-zero, floating-point *Tensor* representing a scale-identity-matrix added to *scale*. May have shape *[B1, ..., Bb]*, *b >= 0*, and characterizes *b*-batches of scale *k x k* identity matrices added to *scale*. When both *scale_identity_multiplier* and *scale_diag* are *None* then *scale* is the *Identity*.
- **skewness** – Skewness parameter. floating-point *Tensor* with shape broadcastable with *event_shape*.
- **tailweight** – Tailweight parameter. floating-point *Tensor* with shape broadcastable with *event_shape*.
- **distribution** – *tf.Distribution*-like instance. Distribution from which *k* iid samples are used as input to transformation *F*. Default is *tfd.Normal(loc=0., scale=1.)*. Must be a scalar-batch, scalar-event distribution. Typically *distribution.reparameterization_type = FULLY_REPARAMETERIZED* or it is a function of non-trainable parameters. WARNING: If you backprop through a VectorSinhArcsinhDiag sample and *distribution* is not *FULLY_REPARAMETERIZED* yet is a function of trainable variables, then the gradient will be incorrect!
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined.
- **name** – Python *str* name prefixed to Ops created by this class.

Raises `ValueError` – if at most *scale_identity_multiplier* is specified.

`inferpy.models.random_variable.`**`VonMises`**`(`**args*, ***kwargs*`)`

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for VonMises.

See VonMises for more details.

Returns RandomVariable.

Original Docstring for Distribution

Construct von Mises distributions with given location and concentration.

The parameters *loc* and *concentration* must be shaped in a way that supports broadcasting (e.g. *loc + concentration* is a valid operation).

Parameters

- **loc** – Floating point tensor, the circular means of the distribution(s).
- **concentration** – Floating point tensor, the level of concentration of the distribution(s) around *loc*. Must take non-negative values. *concentration = 0* defines a Uniform distribution, while *concentration = +inf* indicates a Deterministic distribution at *loc*.
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined.
- **name** – Python *str* name prefixed to Ops created by this class.

Raises `TypeError` – if loc and concentration are different dtypes.

`inferpy.models.random_variable.`**`VonMisesFisher`**`(*args, **kwargs)`

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for VonMisesFisher.

See VonMisesFisher for more details.

Returns RandomVariable.

Original Docstring for Distribution

Creates a new *VonMisesFisher* instance.

Parameters

- **mean_direction** – Floating-point *Tensor* with shape [B1, ... Bn, D]. A unit vector indicating the mode of the distribution, or the unit-normalized direction of the mean. (This is *not* in general the mean of the distribution; the mean is not generally in the support of the distribution.) NOTE: *D* is currently restricted to <= 5.
- **concentration** – Floating-point *Tensor* having batch shape [B1, ... Bn] broadcastable with *mean_direction*. The level of concentration of samples around the *mean_direction*. *concentration=0* indicates a uniform distribution over the unit hypersphere, and *concentration=+inf* indicates a *Deterministic* distribution (delta function) at *mean_direction*.
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined.
- **name** – Python *str* name prefixed to Ops created by this class.

Raises `ValueError` – For known-bad arguments, i.e. unsupported event dimension.

`inferpy.models.random_variable.`**`Wishart`**`(`**args*, ***kwargs*`)`

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for Wishart.

See Wishart for more details.

Returns RandomVariable.

Original Docstring for Distribution

Construct Wishart distributions.

Parameters

- **df** – *float* or *double Tensor*. Degrees of freedom, must be greater than or equal to dimension of the scale matrix.
- **scale** – *float* or *double Tensor*. The symmetric positive definite scale matrix of the distribution. Exactly one of *scale* and 'scale_tril' must be passed.
- **scale_tril** – *float* or *double Tensor*. The Cholesky factorization of the symmetric positive definite scale matrix of the distribution. Exactly one of *scale* and 'scale_tril' must be passed.
- **input_output_cholesky** – Python *bool*. If *True*, functions whose input or output have the semantics of samples assume inputs are in Cholesky form and return outputs in Cholesky form. In particular, if this flag is *True*, input to *log_prob* is presumed of Cholesky form and output from *sample*, *mean*, and *mode* are of Cholesky form. Setting this argument to *True* is purely a computational optimization and does not change the underlying distribution;

for instance, *mean* returns the Cholesky of the mean, not the mean of Cholesky factors. The *variance* and *stddev* methods are unaffected by this flag. Default value: *False* (i.e., input/output does not have Cholesky semantics).

- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined.
- **name** – Python *str* name prefixed to Ops created by this class.

Raises `ValueError` – if zero or both of 'scale' and 'scale_tril' are passed in.

`inferpy.models.random_variable.`**`Zipf`**`(*args, **kwargs)`

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for Zipf.

See Zipf for more details.

Returns RandomVariable.

Original Docstring for Distribution

Initialize a batch of Zipf distributions.

Parameters

- **power** – *Float* like *Tensor* representing the power parameter. Must be strictly greater than *1*.
- **dtype** – The *dtype* of *Tensor* returned by *sample*. Default value: *tf.int32*.
- **interpolate_nondiscrete** – Python *bool*. When *False*, *log_prob* returns *-inf* (and *prob* returns *0*) for non-integer inputs. When *True*, *log_prob* evaluates the continuous function *-power log(k) - log(zeta(power))* , which matches the Zipf pmf at integer arguments *k* (note that this function is not itself a normalized probability log-density). Default value: *True*.
- **sample_maximum_iterations** – Maximum number of iterations of allowable iterations in *sample*. When *validate_args=True*, samples which fail to reach convergence (subject to this cap) are masked out with *self.dtype.min* or *nan* depending on *self.dtype.is_integer*. Default value: *100*.
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs. Default value: *False*.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined. Default value: *False*.

- **name** – Python *str* name prefixed to Ops created by this class. Default value: *'Zipf'*.

Raises `TypeError` – if *power* is not *float* like.

Module contents

`inferpy.models.`**`datamodel`**(*size=None*)

This context is used to declare a plateau model. Random Variables and Parameters will use a sample_shape defined by the argument *size*, or by the *data_model.fit*. If *size* is not specify, the default size 1, or the size specified by *fit* will be used.

`class` `inferpy.models.`**`Parameter`**(*initial_value*, *name=None*)

Bases: `object`

Random Variable parameter which can be optimized by an inference mechanism.

`inferpy.models.`**`probmodel`**(*builder*)

Decorator to create probabilistic models. The function decorated must be a function which declares the Random Variables in the model. It is not needed that the function returns such variables (we capture them using ed.tape).

`inferpy.models.`**`Autoregressive`**(**args*, ***kwargs*)

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for Autoregressive.

See Autoregressive for more details.

Returns RandomVariable.

Original Docstring for Distribution

Construct an *Autoregressive* distribution.

Parameters

- **distribution_fn** – Python *callable* which constructs a *tfd.Distribution*-like instance from a *Tensor* (e.g., *sample0*). The function must respect the "autoregressive property", i.e., there exists a permutation of event such that each coordinate is a diffeomorphic function of on preceding coordinates.
- **sample0** – Initial input to *distribution_fn*; used to build the distribution in *__init__* which in turn specifies this distribution's properties, e.g., *event_shape*, *batch_shape*, *dtype*. If unspecified, then *distribution_fn* should be default constructable.
- **num_steps** – Number of times *distribution_fn* is composed from samples, e.g., *num_steps=2* implies *distribution_fn(distribution_fn(sample0).sample(n)).sample()*.
- **validate_args** – Python *bool*. Whether to validate input with asserts. If *validate_args* is *False*, and the inputs are invalid, correct behavior is not guaranteed.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined.

- **name** – Python *str* name prefixed to Ops created by this class. Default value: "Autoregressive".

Raises

- `ValueError` – if *num_steps* and *num_elements(distribution_fn(sample0).event_shape)* are both *None*.
- `ValueError` – if *num_steps* < *1*.

`inferpy.models.`**`BatchReshape`**`(*args, **kwargs)`

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for BatchReshape.

See BatchReshape for more details.

Returns RandomVariable.

Original Docstring for Distribution

Construct BatchReshape distribution.

Parameters

- **distribution** – The base distribution instance to reshape. Typically an instance of *Distribution*.
- **batch_shape** – Positive *int*-like vector-shaped *Tensor* representing the new shape of the batch dimensions. Up to one dimension may contain *-1*, meaning the remainder of the batch size.
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined.
- **name** – The name to give Ops created by the initializer. Default value: *"BatchReshape"* + *distribution.name*.

Raises

- `ValueError` – if *batch_shape* is not a vector.
- `ValueError` – if *batch_shape* has non-positive elements.
- `ValueError` – if *batch_shape* size is not the same as a *distribution.batch_shape* size.

`inferpy.models.`**`Bernoulli`**`(*args, **kwargs)`

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for Bernoulli.

See Bernoulli for more details.

Returns RandomVariable.

Original Docstring for Distribution

Construct Bernoulli distributions.

Parameters

- **logits** – An N-D *Tensor* representing the log-odds of a *1* event. Each entry in the *Tensor* parametrizes an independent Bernoulli distribution where the probability of an event is sigmoid(logits). Only one of *logits* or *probs* should be passed in.
- **probs** – An N-D *Tensor* representing the probability of a *1* event. Each entry in the *Tensor* parameterizes an independent Bernoulli distribution. Only one of *logits* or *probs* should be passed in.
- **dtype** – The type of the event samples. Default: *int32*.
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined.
- **name** – Python *str* name prefixed to Ops created by this class.

Raises `ValueError` – If p and logits are passed, or if neither are passed.

`inferpy.models.`**`Beta`**`(`*args*, ***kwargs*`)`

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for Beta.

See Beta for more details.

Returns RandomVariable.

Original Docstring for Distribution

Initialize a batch of Beta distributions.

Parameters

- **concentration1** – Positive floating-point *Tensor* indicating mean number of successes; aka "alpha". Implies *self.dtype* and *self.batch_shape*, i.e., *concentration1.shape = [N1, N2, ..., Nm] = self.batch_shape*.
- **concentration0** – Positive floating-point *Tensor* indicating mean number of failures; aka "beta". Otherwise has same semantics as *concentration1*.
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined.
- **name** – Python *str* name prefixed to Ops created by this class.

inferpy.models.**Binomial**(**args*, ***kwargs*)

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for Binomial.

See Binomial for more details.

Returns RandomVariable.

Original Docstring for Distribution

Initialize a batch of Binomial distributions.

Parameters

- **total_count** – Non-negative floating point tensor with shape broadcastable to *[N1,..., Nm]* with *m >= 0* and the same dtype as *probs* or *logits*. Defines this as a batch of *N1 x ... x Nm* different Binomial distributions. Its components should be equal to integer values.
- **logits** – Floating point tensor representing the log-odds of a positive event with shape broadcastable to *[N1,..., Nm] m >= 0*, and the same dtype as *total_count*. Each entry represents logits for the probability of success for independent Binomial distributions. Only one of *logits* or *probs* should be passed in.
- **probs** – Positive floating point tensor with shape broadcastable to *[N1,..., Nm] m >= 0*, *probs in [0, 1]*. Each entry represents the probability of success for independent Binomial distributions. Only one of *logits* or *probs* should be passed in.
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined.
- **name** – Python *str* name prefixed to Ops created by this class.

inferpy.models.**Blockwise**(**args*, ***kwargs*)

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for Blockwise.

See Blockwise for more details.

Returns RandomVariable.

Original Docstring for Distribution

Construct the *Blockwise* distribution.

Parameters

- **distributions** – Python *list* of *tfp.distributions.Distribution* instances. All distribution instances must have the same *batch_shape* and all must have *event_ndims==1*, i.e., be vector-variate distributions.
- **dtype_override** – samples of *distributions* will be cast to this *dtype*. If unspecified, all *distributions* must have the same *dtype*. Default value: *None* (i.e., do not cast).
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined.
- **name** – Python *str* name prefixed to Ops created by this class.

inferpy.models.**Categorical**(**args*, ***kwargs*)

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for Categorical.

See Categorical for more details.

Returns RandomVariable.

Original Docstring for Distribution

Initialize Categorical distributions using class log-probabilities.

Parameters

- **logits** – An N-D *Tensor*, *N >= 1*, representing the log probabilities of a set of Categorical distributions. The first *N - 1* dimensions index into a batch of independent distributions and the last dimension represents a vector of logits for each class. Only one of *logits* or *probs* should be passed in.
- **probs** – An N-D *Tensor*, *N >= 1*, representing the probabilities of a set of Categorical distributions. The first *N - 1* dimensions index into a batch of independent distributions and the last dimension represents a vector of probabilities for each class. Only one of *logits* or *probs* should be passed in.
- **dtype** – The type of the event samples (default: int32).
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined.
- **name** – Python *str* name prefixed to Ops created by this class.

inferpy.models.**Cauchy**(**args*, ***kwargs*)

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for Cauchy.

See Cauchy for more details.

Returns RandomVariable.

Original Docstring for Distribution

Construct Cauchy distributions.

The parameters *loc* and *scale* must be shaped in a way that supports broadcasting (e.g. *loc + scale* is a valid operation).

Parameters

- **loc** – Floating point tensor; the modes of the distribution(s).
- **scale** – Floating point tensor; the locations of the distribution(s). Must contain only positive values.
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined.
- **name** – Python *str* name prefixed to Ops created by this class.

Raises `TypeError` – if *loc* and *scale* have different *dtype*.

`inferpy.models.`**`Chi`**`(*args, **kwargs)`

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for Chi.

See Chi for more details.

Returns RandomVariable.

Original Docstring for Distribution

Construct Chi distributions with parameter *df*.

Parameters

- **`df`** – Floating point tensor, the degrees of freedom of the distribution(s). *df* must contain only positive values.
- **`validate_args`** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **`allow_nan_stats`** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value *NaN* to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined.
- **`name`** – Python *str* name prefixed to Ops created by this class. Default value: *'Chi'*.

`inferpy.models.`**`Chi2`**`(*args, **kwargs)`

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for Chi2.

See Chi2 for more details.

Returns RandomVariable.

Original Docstring for Distribution

Construct Chi2 distributions with parameter *df*.

Parameters

- **`df`** – Floating point tensor, the degrees of freedom of the distribution(s). *df* must contain only positive values.

- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined.
- **name** – Python *str* name prefixed to Ops created by this class.

inferpy.models.**Chi2WithAbsDf**(**args*, ***kwargs*)

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for Chi2WithAbsDf.

See Chi2WithAbsDf for more details.

Returns RandomVariable.

Original Docstring for Distribution

DEPRECATED FUNCTION

Warning: THIS FUNCTION IS DEPRECATED. It will be removed after 2019-06-05. Instructions for updating: Chi2WithAbsDf is deprecated, use Chi2(df=tf.floor(tf.abs(df))) instead.

inferpy.models.**Deterministic**(**args*, ***kwargs*)

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for Deterministic.

See Deterministic for more details.

Returns RandomVariable.

Original Docstring for Distribution

Initialize a scalar *Deterministic* distribution.

The *atol* and *rtol* parameters allow for some slack in *pmf*, *cdf* computations, e.g. due to floating-point error.

``` pmf(x; loc)

= 1, if Abs(x - loc) <= atol + rtol * Abs(loc), = 0, otherwise.

```

Parameters

- **loc** – Numeric *Tensor* of shape *[B1, ..., Bb]*, with *b >= 0*. The point (or batch of points) on which this distribution is supported.
- **atol** – Non-negative *Tensor* of same *dtype* as *loc* and broadcastable shape. The absolute tolerance for comparing closeness to *loc*. Default is *0*.
- **rtol** – Non-negative *Tensor* of same *dtype* as *loc* and broadcastable shape. The relative tolerance for comparing closeness to *loc*. Default is *0*.
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined.
- **name** – Python *str* name prefixed to Ops created by this class.

inferpy.models.**VectorDeterministic**(**args*, ***kwargs*)

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for VectorDeterministic.

See VectorDeterministic for more details.

Returns RandomVariable.

Original Docstring for Distribution

Initialize a *VectorDeterministic* distribution on *R^k*, for *k >= 0*.

Note that there is only one point in *R^0*, the "point" *[]*. So if *k = 0* then *self.prob([]) == 1*.

The *atol* and *rtol* parameters allow for some slack in *pmf* computations, e.g. due to floating-point error.

``` pmf(x; loc)

= 1, if All[Abs(x - loc) <= atol + rtol * Abs(loc)], = 0, otherwise

```

Parameters

- **loc** – Numeric *Tensor* of shape *[B1, ..., Bb, k]*, with *b >= 0*, *k >= 0* The point (or batch of points) on which this distribution is supported.
- **atol** – Non-negative *Tensor* of same *dtype* as *loc* and broadcastable shape. The absolute tolerance for comparing closeness to *loc*. Default is *0*.
- **rtol** – Non-negative *Tensor* of same *dtype* as *loc* and broadcastable shape. The relative tolerance for comparing closeness to *loc*. Default is *0*.
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.

- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined.
- **name** – Python *str* name prefixed to Ops created by this class.

inferpy.models.**Dirichlet**(**args*, ***kwargs*)

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for Dirichlet.

See Dirichlet for more details.

Returns RandomVariable.

Original Docstring for Distribution

Initialize a batch of Dirichlet distributions.

Parameters

- **concentration** – Positive floating-point *Tensor* indicating mean number of class occurrences; aka "alpha". Implies *self.dtype*, and *self.batch_shape*, *self.event_shape*, i.e., if *concentration.shape = [N1, N2, ..., Nm, k]* then *batch_shape = [N1, N2, ..., Nm]* and *event_shape = [k]*.
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined.
- **name** – Python *str* name prefixed to Ops created by this class.

inferpy.models.**DirichletMultinomial**(**args*, ***kwargs*)

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for DirichletMultinomial.

See DirichletMultinomial for more details.

Returns RandomVariable.

Original Docstring for Distribution

Initialize a batch of DirichletMultinomial distributions.

Parameters

- **total_count** – Non-negative floating point tensor, whose dtype is the same as *concentration*. The shape is broadcastable to *[N1,..., Nm]* with *m >= 0*. Defines this as a batch of *N1 x ... x Nm* different Dirichlet multinomial distributions. Its components should be equal to integer values.
- **concentration** – Positive floating point tensor, whose dtype is the same as *n* with shape broadcastable to *[N1,..., Nm, K] m >= 0*. Defines this as a batch of *N1 x ... x Nm* different *K* class Dirichlet multinomial distributions.
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined.
- **name** – Python *str* name prefixed to Ops created by this class.

inferpy.models.**ConditionalDistribution**(**args*, ***kwargs*)

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for ConditionalDistribution.

See ConditionalDistribution for more details.

Returns RandomVariable.

Original Docstring for Distribution

Constructs the *Distribution*.

This is a private method for subclass use.

Parameters

- **dtype** – The type of the event samples. *None* implies no type-enforcement.
- **reparameterization_type** – Instance of *ReparameterizationType*. If *tfd.FULLY_REPARAMETERIZED*, this *Distribution* can be reparameterized in terms of some standard distribution with a function whose Jacobian is constant for the support of the standard distribution. If *tfd.NOT_REPARAMETERIZED*, then no such reparameterization is available.
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.

- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined.
- **parameters** – Python *dict* of parameters used to instantiate this *Distribution*.
- **graph_parents** – Python *list* of graph prerequisites of this *Distribution*.
- **name** – Python *str* name prefixed to Ops created by this class. Default: subclass name.

Raises `ValueError` – if any member of graph_parents is *None* or not a *Tensor*.

`inferpy.models.`**`Distribution`**`(`*args*, ***kwargs*`)`

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for Distribution.

See Distribution for more details.

Returns RandomVariable.

Original Docstring for Distribution

Constructs the *Distribution*.

This is a private method for subclass use.

Parameters

- **dtype** – The type of the event samples. *None* implies no type-enforcement.
- **reparameterization_type** – Instance of *ReparameterizationType*. If *tfd.FULLY_REPARAMETERIZED*, this *Distribution* can be reparameterized in terms of some standard distribution with a function whose Jacobian is constant for the support of the standard distribution. If *tfd.NOT_REPARAMETERIZED*, then no such reparameterization is available.
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined.
- **parameters** – Python *dict* of parameters used to instantiate this *Distribution*.
- **graph_parents** – Python *list* of graph prerequisites of this *Distribution*.
- **name** – Python *str* name prefixed to Ops created by this class. Default: subclass name.

Raises `ValueError` – if any member of graph_parents is *None* or not a *Tensor*.

`inferpy.models.`**`Empirical`**`(`*args*, ***kwargs*`)`

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for Empirical.

See Empirical for more details.

Returns RandomVariable.

Original Docstring for Distribution

Initialize *Empirical* distributions.

Parameters

- **samples** – Numeric *Tensor* of shape [B1, ..., Bk, S, E1, ..., En]', *k, n >= 0*. Samples or batches of samples on which the distribution is based. The first *k* dimensions index into a batch of independent distributions. Length of *S* dimension determines number of samples in each multiset. The last *n* dimension represents samples for each distribution. n is specified by argument event_ndims.
- **event_ndims** – Python *int32*, default *0*. number of dimensions for each event. When *0* this distribution has scalar samples. When *1* this distribution has vector-like samples.
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value *NaN* to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined.
- **name** – Python *str* name prefixed to Ops created by this class.

Raises `ValueError` – if the rank of *samples* < event_ndims + 1.

`inferpy.models.`**`Exponential`**`(*args, **kwargs)`

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for Exponential.

See Exponential for more details.

Returns RandomVariable.

Original Docstring for Distribution

Construct Exponential distribution with parameter *rate*.

Parameters

- **rate** – Floating point tensor, equivalent to *1 / mean*. Must contain only positive values.

- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined.
- **name** – Python *str* name prefixed to Ops created by this class.

inferpy.models.**FiniteDiscrete**(**args*, ***kwargs*)

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for FiniteDiscrete.

See FiniteDiscrete for more details.

Returns RandomVariable.

Original Docstring for Distribution

Construct a finite discrete contribution.

Parameters

- **outcomes** – A 1-D floating or integer *Tensor*, representing a list of possible outcomes in strictly ascending order.
- **logits** – A floating N-D *Tensor*, *N >= 1*, representing the log probabilities of a set of FiniteDiscrete distributions. The first *N - 1* dimensions index into a batch of independent distributions and the last dimension represents a vector of logits for each discrete value. Only one of *logits* or *probs* should be passed in.
- **probs** – A floating N-D *Tensor*, *N >= 1*, representing the probabilities of a set of FiniteDiscrete distributions. The first *N - 1* dimensions index into a batch of independent distributions and the last dimension represents a vector of probabilities for each discrete value. Only one of *logits* or *probs* should be passed in.
- **rtol** – *Tensor* with same *dtype* as *outcomes*. The relative tolerance for floating number comparison. Only effective when *outcomes* is a floating *Tensor*. Default is *10 * eps*.
- **atol** – *Tensor* with same *dtype* as *outcomes*. The absolute tolerance for floating number comparison. Only effective when *outcomes* is a floating *Tensor*. Default is *10 * eps*.
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may render incorrect outputs.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value '*NaN*' to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined.
- **name** – Python *str* name prefixed to Ops created by this class.

inferpy.models.**Gamma**(**args*, ***kwargs*)

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for Gamma.

See Gamma for more details.

Returns RandomVariable.

Original Docstring for Distribution

Construct Gamma with *concentration* and *rate* parameters.

The parameters *concentration* and *rate* must be shaped in a way that supports broadcasting (e.g. *concentration* + *rate* is a valid operation).

Parameters

- **concentration** – Floating point tensor, the concentration params of the distribution(s). Must contain only positive values.
- **rate** – Floating point tensor, the inverse scale params of the distribution(s). Must contain only positive values.
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined.
- **name** – Python *str* name prefixed to Ops created by this class.

Raises TypeError – if *concentration* and *rate* are different dtypes.

inferpy.models.**GammaGamma**(**args*, ***kwargs*)

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for GammaGamma.

See GammaGamma for more details.

Returns RandomVariable.

Original Docstring for Distribution

Initializes a batch of Gamma-Gamma distributions.

The parameters *concentration* and *rate* must be shaped in a way that supports broadcasting (e.g. *concentration + mixing_concentration + mixing_rate* is a valid operation).

Parameters

- **concentration** – Floating point tensor, the concentration params of the distribution(s). Must contain only positive values.
- **mixing_concentration** – Floating point tensor, the concentration params of the mixing Gamma distribution(s). Must contain only positive values.
- **mixing_rate** – Floating point tensor, the rate params of the mixing Gamma distribution(s). Must contain only positive values.
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined.
- **name** – Python *str* name prefixed to Ops created by this class.

Raises `TypeError` – if *concentration* and *rate* are different dtypes.

`inferpy.models.`**`GaussianProcess`**`(`**args*, ***kwargs*`)`

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for GaussianProcess.

See GaussianProcess for more details.

Returns RandomVariable.

Original Docstring for Distribution

Instantiate a GaussianProcess Distribution.

Parameters

- **kernel** – *PositiveSemidefiniteKernel*-like instance representing the GP's covariance function.
- **index_points** – *float Tensor* representing finite (batch of) vector(s) of points in the index set over which the GP is defined. Shape has the form *[b1, ..., bB, e, f1, ..., fF]* where *F* is the number of feature dimensions and must equal *kernel.feature_ndims* and *e* is the number (size) of index points in each batch. Ultimately this distribution corresponds to a *e*-dimensional multivariate normal. The batch shape must be broadcastable with *kernel.batch_shape* and any batch dims yielded by *mean_fn*.
- **mean_fn** – Python *callable* that acts on *index_points* to produce a (batch of) vector(s) of mean values at *index_points*. Takes a *Tensor* of shape *[b1, ..., bB, f1, ..., fF]* and returns a *Tensor* whose shape is broadcastable with *[b1, ..., bB]*. Default value: *None* implies constant zero function.

- **observation_noise_variance** – *float Tensor* representing the variance of the noise in the Normal likelihood distribution of the model. May be batched, in which case the batch shape must be broadcastable with the shapes of all other batched parameters (*kernel.batch_shape*, *index_points*, etc.). Default value: *0.*
- **jitter** – *float* scalar *Tensor* added to the diagonal of the covariance matrix to ensure positive definiteness of the covariance matrix. Default value: *1e-6*.
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs. Default value: *False*.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined. Default value: *False*.
- **name** – Python *str* name prefixed to Ops created by this class. Default value: "GaussianProcess".

Raises `ValueError` – if *mean_fn* is not *None* and is not callable.

`inferpy.models.`**`GaussianProcessRegressionModel`**`(*args, **kwargs)`

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for GaussianProcessRegressionModel.

See GaussianProcessRegressionModel for more details.

Returns RandomVariable.

Original Docstring for Distribution

Construct a GaussianProcessRegressionModel instance.

Parameters

- **kernel** – *PositiveSemidefiniteKernel*-like instance representing the GP's covariance function.
- **index_points** – *float Tensor* representing finite collection, or batch of collections, of points in the index set over which the GP is defined. Shape has the form *[b1, ..., bB, e, f1, ..., fF]* where *F* is the number of feature dimensions and must equal *kernel.feature_ndims* and *e* is the number (size) of index points in each batch. Ultimately this distribution corresponds to an *e*-dimensional multivariate normal. The batch shape must be broadcastable with *kernel.batch_shape* and any batch dims yielded by *mean_fn*.
- **observation_index_points** – *float Tensor* representing finite collection, or batch of collections, of points in the index set for which some data has been observed. Shape has the form *[b1, ..., bB, e, f1, ..., fF]* where *F* is the number of feature dimensions and must equal *kernel.feature_ndims*, and *e* is the number (size) of index points in each batch. *[b1, ..., bB, e]* must be broadcastable with the shape of *observations*, and *[b1, ..., bB]* must be broadcastable with the shapes of all other batched parameters (*kernel.batch_shape*,

index_points, etc). The default value is *None*, which corresponds to the empty set of observations, and simply results in the prior predictive model (a GP with noise of variance *predictive_noise_variance*).

- **observations** – *float Tensor* representing collection, or batch of collections, of observations corresponding to *observation_index_points*. Shape has the form *[b1, ..., bB, e]*, which must be brodcastable with the batch and example shapes of *observation_index_points*. The batch shape *[b1, ..., bB]* must be broadcastable with the shapes of all other batched parameters (*kernel.batch_shape*, *index_points*, etc.). The default value is *None*, which corresponds to the empty set of observations, and simply results in the prior predictive model (a GP with noise of variance *predictive_noise_variance*).
- **observation_noise_variance** – *float Tensor* representing the variance of the noise in the Normal likelihood distribution of the model. May be batched, in which case the batch shape must be broadcastable with the shapes of all other batched parameters (*kernel.batch_shape*, *index_points*, etc.). Default value: *0.*
- **predictive_noise_variance** – *float Tensor* representing the variance in the posterior predictive model. If *None*, we simply re-use *observation_noise_variance* for the posterior predictive noise. If set explicitly, however, we use this value. This allows us, for example, to omit predictive noise variance (by setting this to zero) to obtain noiseless posterior predictions of function values, conditioned on noisy observations.
- **mean_fn** – Python *callable* that acts on *index_points* to produce a collection, or batch of collections, of mean values at *index_points*. Takes a *Tensor* of shape *[b1, ..., bB, f1, ..., fF]* and returns a *Tensor* whose shape is broadcastable with *[b1, ..., bB]*. Default value: *None* implies the constant zero function.
- **jitter** – *float* scalar *Tensor* added to the diagonal of the covariance matrix to ensure positive definiteness of the covariance matrix. Default value: *1e-6*.
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs. Default value: *False*.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value *NaN* to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined. Default value: *False*.
- **name** – Python *str* name prefixed to Ops created by this class. Default value: 'GaussianProcessRegressionModel'.

Raises `ValueError` – if either - only one of *observations* and *observation_index_points* is given, or - *mean_fn* is not *None* and not callable.

`inferpy.models.`**`Geometric`**`(*args, **kwargs)`

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for Geometric.

See Geometric for more details.

Returns RandomVariable.

Original Docstring for Distribution

Construct Geometric distributions.

Parameters

- **logits** – Floating-point *Tensor* with shape *[B1, ..., Bb]* where *b >= 0* indicates the number of batch dimensions. Each entry represents logits for the probability of success for independent Geometric distributions and must be in the range *(-inf, inf]*. Only one of *logits* or *probs* should be specified.
- **probs** – Positive floating-point *Tensor* with shape *[B1, ..., Bb]* where *b >= 0* indicates the number of batch dimensions. Each entry represents the probability of success for independent Geometric distributions and must be in the range *(0, 1]*. Only one of *logits* or *probs* should be specified.
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined.
- **name** – Python *str* name prefixed to Ops created by this class.

inferpy.models.**Gumbel**(**args*, ***kwargs*)

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for Gumbel.

See Gumbel for more details.

Returns RandomVariable.

Original Docstring for Distribution

Construct Gumbel distributions with location and scale *loc* and *scale*.

The parameters *loc* and *scale* must be shaped in a way that supports broadcasting (e.g. *loc + scale* is a valid operation).

Parameters

- **loc** – Floating point tensor, the means of the distribution(s).
- **scale** – Floating point tensor, the scales of the distribution(s). scale must contain only positive values.
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs. Default value: *False*.

- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined. Default value: *True*.
- **name** – Python *str* name prefixed to Ops created by this class. Default value: *'Gumbel'*.

Raises TypeError – if loc and scale are different dtypes.

inferpy.models.**HalfCauchy**(**args*, ***kwargs*)

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for HalfCauchy.

See HalfCauchy for more details.

Returns RandomVariable.

Original Docstring for Distribution

Construct a half-Cauchy distribution with *loc* and *scale*.

Parameters

- **loc** – Floating-point *Tensor*; the location(s) of the distribution(s).
- **scale** – Floating-point *Tensor*; the scale(s) of the distribution(s). Must contain only positive values.
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs. Default value: *False* (i.e. do not validate args).
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined. Default value: *True*.
- **name** – Python *str* name prefixed to Ops created by this class. Default value: 'HalfCauchy'.

Raises TypeError – if *loc* and *scale* have different *dtype*.

inferpy.models.**HalfNormal**(**args*, ***kwargs*)

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for HalfNormal.

See HalfNormal for more details.

Returns RandomVariable.

Original Docstring for Distribution

Construct HalfNormals with scale *scale*.

Parameters

- **scale** – Floating point tensor; the scales of the distribution(s). Must contain only positive values.
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined.
- **name** – Python *str* name prefixed to Ops created by this class.

inferpy.models.**HiddenMarkovModel**(**args*, ***kwargs*)

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for HiddenMarkovModel.

See HiddenMarkovModel for more details.

Returns RandomVariable.

Original Docstring for Distribution

Initialize hidden Markov model.

Parameters

- **initial_distribution** – A *Categorical*-like instance. Determines probability of first hidden state in Markov chain. The number of categories must match the number of categories of *transition_distribution* as well as both the rightmost batch dimension of *transition_distribution* and the rightmost batch dimension of *observation_distribution*.
- **transition_distribution** – A *Categorical*-like instance. The rightmost batch dimension indexes the probability distribution of each hidden state conditioned on the previous hidden state.
- **observation_distribution** – A *tfp.distributions.Distribution*-like instance. The rightmost batch dimension indexes the distribution of each observation conditioned on the corresponding hidden state.
- **num_steps** – The number of steps taken in Markov chain. A python *int*.
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs. Default value: *False*.

- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined. Default value: *True*.
- **name** – Python *str* name prefixed to Ops created by this class. Default value: "HiddenMarkovModel".

Raises

- `ValueError` – if *num_steps* is not at least 1.
- `ValueError` – if *initial_distribution* does not have scalar *event_shape*.
- `ValueError` – if *transition_distribution* does not have scalar *event_shape.*
- `ValueError` – if *transition_distribution* and *observation_distribution* are fully defined but don't have matching rightmost dimension.

`inferpy.models.`**`Horseshoe`**`(*args, **kwargs)`

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for Horseshoe.

See Horseshoe for more details.

Returns RandomVariable.

Original Docstring for Distribution

Construct a Horseshoe distribution with *scale*.

Parameters

- **scale** – Floating point tensor; the scales of the distribution(s). Must contain only positive values.
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs. Default value: *False* (i.e., do not validate args).
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined. Default value: *True*.
- **name** – Python *str* name prefixed to Ops created by this class. Default value: 'Horseshoe'.

`inferpy.models.`**`Independent`**`(*args, **kwargs)`

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for Independent.

See Independent for more details.

Returns RandomVariable.

Original Docstring for Distribution

Construct a *Independent* distribution.

Parameters

- **distribution** – The base distribution instance to transform. Typically an instance of *Distribution*.
- **reinterpreted_batch_ndims** – Scalar, integer number of rightmost batch dims which will be regarded as event dims. When *None* all but the first batch axis (batch axis 0) will be transferred to event dimensions (analogous to *tf.layers.flatten*).
- **validate_args** – Python *bool*. Whether to validate input with asserts. If *validate_args* is *False*, and the inputs are invalid, correct behavior is not guaranteed.
- **name** – The name for ops managed by the distribution. Default value: *Independent* + *distribution.name*.

Raises `ValueError` – if *reinterpreted_batch_ndims* exceeds *distribution.batch_ndims*

`inferpy.models.`**`InverseGamma`**`(*args, **kwargs)`

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for InverseGamma.

See InverseGamma for more details.

Returns RandomVariable.

Original Docstring for Distribution

Construct InverseGamma with *concentration* and *scale* parameters. (deprecated arguments)

Warning: SOME ARGUMENTS ARE DEPRECATED: *(rate)*. They will be removed after 2019-05-08. Instructions for updating: The *rate* parameter is deprecated. Use *scale* instead.The *rate* parameter was always interpreted as a *scale* parameter, but erroneously misnamed.

The parameters *concentration* and *scale* must be shaped in a way that supports broadcasting (e.g. *concentration* + *scale* is a valid operation).

Parameters

- **concentration** – Floating point tensor, the concentration params of the distribution(s). Must contain only positive values.
- **scale** – Floating point tensor, the scale params of the distribution(s). Must contain only positive values.

- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined.
- **rate** – Deprecated (mis-named) alias for *scale*.
- **name** – Python *str* name prefixed to Ops created by this class.

Raises `TypeError` – if *concentration* and *scale* are different dtypes.

`inferpy.models.`**`InverseGaussian`**(**args*, ***kwargs*)

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for InverseGaussian.

See InverseGaussian for more details.

Returns RandomVariable.

Original Docstring for Distribution

Constructs inverse Gaussian distribution with *loc* and *concentration*.

Parameters

- **loc** – Floating-point *Tensor*, the loc params. Must contain only positive values.
- **concentration** – Floating-point *Tensor*, the concentration params. Must contain only positive values.
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs. Default value: *False* (i.e. do not validate args).
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined. Default value: *True*.
- **name** – Python *str* name prefixed to Ops created by this class. Default value: 'InverseGaussian'.

`inferpy.models.`**`JointDistribution`**(**args*, ***kwargs*)

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for JointDistribution.

See JointDistribution for more details.

Returns RandomVariable.

Original Docstring for Distribution

Constructs the *Distribution*.

This is a private method for subclass use.

Parameters

- **dtype** – The type of the event samples. *None* implies no type-enforcement.
- **reparameterization_type** – Instance of *ReparameterizationType*. If *tfd.FULLY_REPARAMETERIZED*, this *Distribution* can be reparameterized in terms of some standard distribution with a function whose Jacobian is constant for the support of the standard distribution. If *tfd.NOT_REPARAMETERIZED*, then no such reparameterization is available.
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined.
- **parameters** – Python *dict* of parameters used to instantiate this *Distribution*.
- **graph_parents** – Python *list* of graph prerequisites of this *Distribution*.
- **name** – Python *str* name prefixed to Ops created by this class. Default: subclass name.

Raises `ValueError` – if any member of graph_parents is *None* or not a *Tensor*.

`inferpy.models.`**`JointDistributionCoroutine`**`(`**args*, ***kwargs*`)`

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for JointDistributionCoroutine.

See JointDistributionCoroutine for more details.

Returns RandomVariable.

Original Docstring for Distribution

Construct the *JointDistributionCoroutine* distribution.

Parameters

- **model** – A generator that yields a sequence of *tfd.Distribution*-like instances.

- **sample_dtype** – Samples from this distribution will be structured like *tf.nest.pack_sequence_as(sample_dtype, list_)*. *sample_dtype* is only used for *tf.nest.pack_sequence_as* structuring of outputs, never casting (which is the responsibility of the component distributions). Default value: *None* (i.e., *tuple*).
- **validate_args** – Python *bool*. Whether to validate input with asserts. If *validate_args* is *False*, and the inputs are invalid, correct behavior is not guaranteed. Default value: *False*.
- **name** – The name for ops managed by the distribution. Default value: *None* (i.e., *"JointDistributionCoroutine"*).

inferpy.models.**JointDistributionNamed**(**args*, ***kwargs*)

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for JointDistributionNamed.

See JointDistributionNamed for more details.

Returns RandomVariable.

Original Docstring for Distribution

Construct the *JointDistributionNamed* distribution.

Parameters

- **model** – Python *dict* or *namedtuple* of distribution-making functions each with required args corresponding only to other keys.
- **validate_args** – Python *bool*. Whether to validate input with asserts. If *validate_args* is *False*, and the inputs are invalid, correct behavior is not guaranteed. Default value: *False*.
- **name** – The name for ops managed by the distribution. Default value: *None* (i.e., *"JointDistributionNamed"*).

inferpy.models.**JointDistributionSequential**(**args*, ***kwargs*)

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for JointDistributionSequential.

See JointDistributionSequential for more details.

Returns RandomVariable.

Original Docstring for Distribution

Construct the *JointDistributionSequential* distribution.

Parameters

- **model** – Python list of either tfd.Distribution instances and/or lambda functions which take the *k* previous distributions and returns a new tfd.Distribution instance.
- **validate_args** – Python *bool*. Whether to validate input with asserts. If *validate_args* is *False*, and the inputs are invalid, correct behavior is not guaranteed. Default value: *False*.
- **name** – The name for ops managed by the distribution. Default value: *None* (i.e., *"JointDistributionSequential"*).

inferpy.models.**Kumaraswamy**(**args*, ***kwargs*)

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for Kumaraswamy.

See Kumaraswamy for more details.

Returns RandomVariable.

Original Docstring for Distribution

Initialize a batch of Kumaraswamy distributions.

Parameters

- **concentration1** – Positive floating-point *Tensor* indicating mean number of successes; aka "alpha". Implies *self.dtype* and *self.batch_shape*, i.e., *concentration1.shape = [N1, N2, ..., Nm] = self.batch_shape*.
- **concentration0** – Positive floating-point *Tensor* indicating mean number of failures; aka "beta". Otherwise has same semantics as *concentration1*.
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined.
- **name** – Python *str* name prefixed to Ops created by this class.

inferpy.models.**Laplace**(**args*, ***kwargs*)

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for Laplace.

See Laplace for more details.

Returns RandomVariable.

Original Docstring for Distribution

Construct Laplace distribution with parameters *loc* and *scale*.

The parameters *loc* and *scale* must be shaped in a way that supports broadcasting (e.g., *loc / scale* is a valid operation).

Parameters

- **loc** – Floating point tensor which characterizes the location (center) of the distribution.
- **scale** – Positive floating point tensor which characterizes the spread of the distribution.
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined.
- **name** – Python *str* name prefixed to Ops created by this class.

Raises `TypeError` – if *loc* and *scale* are of different dtype.

`inferpy.models.`**`LinearGaussianStateSpaceModel`**`(`*args*, ***kwargs*`)`

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for LinearGaussianStateSpaceModel.

See LinearGaussianStateSpaceModel for more details.

Returns RandomVariable.

Original Docstring for Distribution

Initialize a 'LinearGaussianStateSpaceModel.

Parameters

- **num_timesteps** – Integer *Tensor* total number of timesteps.
- **transition_matrix** – A transition operator, represented by a Tensor or LinearOperator of shape *[latent_size, latent_size]*, or by a callable taking as argument a scalar integer Tensor *t* and returning a Tensor or LinearOperator representing the transition operator from latent state at time *t* to time *t + 1*.
- **transition_noise** – An instance of *tfd.MultivariateNormalLinearOperator* with event shape *[latent_size]*, representing the mean and covariance of the transition noise model, or a callable taking as argument a scalar integer Tensor *t* and returning such a distribution representing the noise in the transition from time *t* to time *t + 1*.

- **observation_matrix** – An observation operator, represented by a Tensor or LinearOperator of shape *[observation_size, latent_size]*, or by a callable taking as argument a scalar integer Tensor *t* and returning a timestep-specific Tensor or LinearOperator.
- **observation_noise** – An instance of *tfd.MultivariateNormalLinearOperator* with event shape *[observation_size]*, representing the mean and covariance of the observation noise model, or a callable taking as argument a scalar integer Tensor *t* and returning a timestep-specific noise model.
- **initial_state_prior** – An instance of *MultivariateNormalLinearOperator* representing the prior distribution on latent states; must have event shape *[latent_size]*.
- **initial_step** – optional *int* specifying the time of the first modeled timestep. This is added as an offset when passing timesteps *t* to (optional) callables specifying timestep-specific transition and observation models.
- **validate_args** – Python *bool*, default *False*. Whether to validate input with asserts. If *validate_args* is *False*, and the inputs are invalid, correct behavior is not guaranteed.
- **allow_nan_stats** – Python *bool*, default *True*. If *False*, raise an exception if a statistic (e.g. mean/mode/etc...) is undefined for any batch member If *True*, batch members with valid parameters leading to undefined statistics will return NaN for this statistic.
- **name** – The name to give Ops created by the initializer.

inferpy.models.**LKJ**(**args*, ***kwargs*)

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for LKJ.

See LKJ for more details.

Returns RandomVariable.

Original Docstring for Distribution

Construct LKJ distributions.

Parameters

- **dimension** – Python *int*. The dimension of the correlation matrices to sample.
- **concentration** – *float* or *double Tensor*. The positive concentration parameter of the LKJ distributions. The pdf of a sample matrix *X* is proportional to *det(X) ** (concentration - 1)*.
- **input_output_cholesky** – Python *bool*. If *True*, functions whose input or output have the semantics of samples assume inputs are in Cholesky form and return outputs in Cholesky form. In particular, if this flag is *True*, input to *log_prob* is presumed of Cholesky form and output from *sample* is of Cholesky form. Setting this argument to *True* is purely a computational optimization and does not change the underlying distribution. Additionally, validation checks which are only defined on the multiplied-out form are omitted, even if *validate_args* is *True*. Default value: *False* (i.e., input/output does not have Cholesky semantics).

- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value *NaN* to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined.
- **name** – Python *str* name prefixed to Ops created by this class.

Raises `ValueError` – If *dimension* is negative.

`inferpy.models.`**`Logistic`**`(`*args*, ***kwargs*`)`

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for Logistic.

See Logistic for more details.

Returns RandomVariable.

Original Docstring for Distribution

Construct Logistic distributions with mean and scale *loc* and *scale*.

The parameters *loc* and *scale* must be shaped in a way that supports broadcasting (e.g. *loc* + *scale* is a valid operation).

Parameters

- **loc** – Floating point tensor, the means of the distribution(s).
- **scale** – Floating point tensor, the scales of the distribution(s). Must contain only positive values.
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined.
- **name** – The name to give Ops created by the initializer.

Raises `TypeError` – if loc and scale are different dtypes.

`inferpy.models.`**`LogNormal`**`(`*args*, ***kwargs*`)`

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.

- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for LogNormal.

See LogNormal for more details.

Returns RandomVariable.

Original Docstring for Distribution

Construct a log-normal distribution.

The LogNormal distribution models positive-valued random variables whose logarithm is normally distributed with mean *loc* and standard deviation *scale*. It is constructed as the exponential transformation of a Normal distribution.

Parameters

- **loc** – Floating-point *Tensor*; the means of the underlying Normal distribution(s).
- **scale** – Floating-point *Tensor*; the stddevs of the underlying Normal distribution(s).
- **validate_args** – Python *bool*, default *False*. Whether to validate input with asserts. If *validate_args* is *False*, and the inputs are invalid, correct behavior is not guaranteed.
- **allow_nan_stats** – Python *bool*, default *True*. If *False*, raise an exception if a statistic (e.g. mean/mode/etc...) is undefined for any batch member If *True*, batch members with valid parameters leading to undefined statistics will return NaN for this statistic.
- **name** – The name to give Ops created by the initializer.

inferpy.models.**Mixture**(**args*, ***kwargs*)

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for Mixture.

See Mixture for more details.

Returns RandomVariable.

Original Docstring for Distribution

Initialize a Mixture distribution.

A *Mixture* is defined by a *Categorical* (*cat*, representing the mixture probabilities) and a list of *Distribution* objects all having matching dtype, batch shape, event shape, and continuity properties (the components).

The *num_classes* of *cat* must be possible to infer at graph construction time and match *len(components)*.

Parameters

- **cat** – A *Categorical* distribution instance, representing the probabilities of *distributions*.
- **components** – A list or tuple of *Distribution* instances. Each instance must have the same type, be defined on the same domain, and have matching *event_shape* and *batch_shape*.

- **validate_args** – Python *bool*, default *False*. If *True*, raise a runtime error if batch or event ranks are inconsistent between cat and any of the distributions. This is only checked if the ranks cannot be determined statically at graph construction time.
- **allow_nan_stats** – Boolean, default *True*. If *False*, raise an exception if a statistic (e.g. mean/mode/etc...) is undefined for any

  batch member. If *True*, batch members with valid parameters leading to undefined statistics will return NaN for this statistic.
- **use_static_graph** – Calls to *sample* will not rely on dynamic tensor indexing, allowing for some static graph compilation optimizations, but at the expense of sampling all underlying distributions in the mixture. (Possibly useful when running on TPUs). Default value: *False* (i.e., use dynamic indexing).
- **name** – A name for this distribution (optional).

Raises

- `TypeError` – If cat is not a *Categorical*, or *components* is not a list or tuple, or the elements of *components* are not instances of *Distribution*, or do not have matching *dtype*.
- `ValueError` – If *components* is an empty list or tuple, or its elements do not have a statically known event rank. If *cat.num_classes* cannot be inferred at graph creation time, or the constant value of *cat.num_classes* is not equal to *len(components)*, or all *components* and *cat* do not have matching static batch shapes, or all components do not have matching static event shapes.

`inferpy.models.`**`MixtureSameFamily`**(**args*, ***kwargs*)

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for MixtureSameFamily.

See MixtureSameFamily for more details.

Returns RandomVariable.

Original Docstring for Distribution

Construct a *MixtureSameFamily* distribution.

Parameters

- **mixture_distribution** – *tfp.distributions.Categorical*-like instance. Manages the probability of selecting components. The number of categories must match the rightmost batch dimension of the *components_distribution*. Must have either scalar *batch_shape* or *batch_shape* matching *components_distribution.batch_shape[:-1]*.
- **components_distribution** – *tfp.distributions.Distribution*-like instance. Right-most batch dimension indexes components.
- **reparameterize** – Python *bool*, default *False*. Whether to reparameterize samples of the distribution using implicit reparameterization gradients [(Figurnov et al., 2018)][1]. The gradients for the mixture logits are equivalent to the ones described by [(Graves, 2016)][2].

The gradients for the components parameters are also computed using implicit reparameterization (as opposed to ancestral sampling), meaning that all components are updated every step. Only works when:

(1) components_distribution is fully reparameterized;

(2) components_distribution is either a scalar distribution or fully factorized (tfd.Independent applied to a scalar distribution); (3) batch shape has a known rank.

Experimental, may be slow and produce infs/NaNs.

- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined.
- **name** – Python *str* name prefixed to Ops created by this class.

Raises

- `ValueError` – *if not dtype_util.is_integer(mixture_distribution.dtype)*.
- `ValueError` – if mixture_distribution does not have scalar *event_shape*.
- `ValueError` – if *mixture_distribution.batch_shape* and *components_distribution.batch_shape[:-1]* are both fully defined and the former is neither scalar nor equal to the latter.
- `ValueError` – if *mixture_distribution* categories does not equal *components_distribution* rightmost batch shape.

References

[1]: Michael Figurnov, Shakir Mohamed and Andriy Mnih. Implicit reparameterization gradients. In _Neural Information Processing Systems_, 2018. https://arxiv.org/abs/1805.08498

[2]: Alex Graves. Stochastic Backpropagation through Mixture Density Distributions. _arXiv_, 2016. https://arxiv.org/abs/1607.05690

`inferpy.models.`**`Multinomial`**`(*args, **kwargs)`

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for Multinomial.

See Multinomial for more details.

Returns RandomVariable.

Original Docstring for Distribution

Initialize a batch of Multinomial distributions.

Parameters

- **total_count** – Non-negative floating point tensor with shape broadcastable to *[N1,..., Nm]* with *m >= 0*. Defines this as a batch of *N1 x ... x Nm* different Multinomial distributions. Its components should be equal to integer values.
- **logits** – Floating point tensor representing unnormalized log-probabilities of a positive event with shape broadcastable to *[N1,..., Nm, K] m >= 0*, and the same dtype as *total_count*. Defines this as a batch of *N1 x ... x Nm* different *K* class Multinomial distributions. Only one of *logits* or *probs* should be passed in.
- **probs** – Positive floating point tensor with shape broadcastable to *[N1,..., Nm, K] m >= 0* and same dtype as *total_count*. Defines this as a batch of *N1 x ... x Nm* different *K* class Multinomial distributions. *probs*'s components in the last portion of its shape should sum to *1*. Only one of *logits* or *probs* should be passed in.
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined.
- **name** – Python *str* name prefixed to Ops created by this class.

inferpy.models.**MultivariateStudentTLinearOperator**(**args*, ***kwargs*)

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for MultivariateStudentTLinearOperator.

See MultivariateStudentTLinearOperator for more details.

Returns RandomVariable.

Original Docstring for Distribution

Construct Multivariate Student's t-distribution on *R^k*.

The *batch_shape* is the broadcast shape between *df*, *loc* and *scale* arguments.

The *event_shape* is given by last dimension of the matrix implied by *scale*. The last dimension of *loc* must broadcast with this.

Additional leading dimensions (if any) will index batches.

Parameters

- **df** – A positive floating-point *Tensor*. Has shape *[B1, ..., Bb]* where *b >= 0*.
- **loc** – Floating-point *Tensor*. Has shape *[B1, ..., Bb, k]* where *k* is the event size.
- **scale** – Instance of *LinearOperator* with a floating *dtype* and shape *[B1, ..., Bb, k, k]*.
- **validate_args** – Python *bool*, default *False*. Whether to validate input with asserts. If *validate_args* is *False*, and the inputs are invalid, correct behavior is not guaranteed.

- **allow_nan_stats** – Python *bool*, default *True*. If *False*, raise an exception if a statistic (e.g. mean/variance/etc...) is undefined for any batch member If *True*, batch members with valid parameters leading to undefined statistics will return NaN for this statistic.
- **name** – The name to give Ops created by the initializer.

Raises

- `TypeError` – if not *scale.dtype.is_floating*.
- `ValueError` – if not *scale.is_positive_definite*.

`inferpy.models.`**`MultivariateNormalDiag`**`(*args, **kwargs)`

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for MultivariateNormalDiag.

See MultivariateNormalDiag for more details.

Returns RandomVariable.

Original Docstring for Distribution

Construct Multivariate Normal distribution on *R^k*.

The *batch_shape* is the broadcast shape between *loc* and *scale* arguments.

The *event_shape* is given by last dimension of the matrix implied by *scale*. The last dimension of *loc* (if provided) must broadcast with this.

Recall that *covariance = scale @ scale.T*. A (non-batch) *scale* matrix is:

```
`none scale = diag(scale_diag + scale_identity_multiplier * ones(k)) `
```

where:

- *scale_diag.shape = [k]*, and,
- *scale_identity_multiplier.shape = []*.

Additional leading dimensions (if any) will index batches.

If both *scale_diag* and *scale_identity_multiplier* are *None*, then *scale* is the Identity matrix.

Parameters

- **loc** – Floating-point *Tensor*. If this is set to *None*, *loc* is implicitly *0*. When specified, may have shape *[B1, ..., Bb, k]* where *b >= 0* and *k* is the event size.
- **scale_diag** – Non-zero, floating-point *Tensor* representing a diagonal matrix added to *scale*. May have shape *[B1, ..., Bb, k]*, *b >= 0*, and characterizes *b*-batches of *k x k* diagonal matrices added to *scale*. When both *scale_identity_multiplier* and *scale_diag* are *None* then *scale* is the *Identity*.
- **scale_identity_multiplier** – Non-zero, floating-point *Tensor* representing a scaled-identity-matrix added to *scale*. May have shape *[B1, ..., Bb]*, *b >= 0*, and

characterizes *b*-batches of scaled *k x k* identity matrices added to *scale*. When both *scale_identity_multiplier* and *scale_diag* are *None* then *scale* is the *Identity*.

- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined.
- **name** – Python *str* name prefixed to Ops created by this class.

Raises ValueError – if at most *scale_identity_multiplier* is specified.

inferpy.models.**MultivariateNormalDiagWithSoftplusScale**(**args*, ***kwargs*)

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for MultivariateNormalDiagWithSoftplusScale.

See MultivariateNormalDiagWithSoftplusScale for more details.

Returns RandomVariable.

Original Docstring for Distribution

DEPRECATED FUNCTION

Warning: THIS FUNCTION IS DEPRECATED. It will be removed after 2019-06-05. Instructions for updating: MultivariateNormalDiagWithSoftplusScale is deprecated, use MultivariateNormalDiag(loc=loc, scale_diag=tf.nn.softplus(scale_diag)) instead.

inferpy.models.**MultivariateNormalDiagPlusLowRank**(**args*, ***kwargs*)

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for MultivariateNormalDiagPlusLowRank.

See MultivariateNormalDiagPlusLowRank for more details.

Returns RandomVariable.

Original Docstring for Distribution

Construct Multivariate Normal distribution on *R^k*.

The *batch_shape* is the broadcast shape between *loc* and *scale* arguments.

The *event_shape* is given by last dimension of the matrix implied by *scale*. The last dimension of *loc* (if provided) must broadcast with this.

Recall that *covariance* = *scale* @ *scale.T*. A (non-batch) *scale* matrix is:

‘‘‘none scale = diag(scale_diag + scale_identity_multiplier ones(k)) +

scale_perturb_factor @ diag(scale_perturb_diag) @ scale_perturb_factor.T

‘‘‘

where:

- *scale_diag.shape* = *[k]*,
- *scale_identity_multiplier.shape* = *[]*,
- *scale_perturb_factor.shape* = *[k, r]*, typically *k* >> *r*, and,
- *scale_perturb_diag.shape* = *[r]*.

Additional leading dimensions (if any) will index batches.

If both *scale_diag* and *scale_identity_multiplier* are *None*, then *scale* is the Identity matrix.

Parameters

- **loc** – Floating-point *Tensor*. If this is set to *None*, *loc* is implicitly *0*. When specified, may have shape *[B1, ..., Bb, k]* where *b* >= *0* and *k* is the event size.
- **scale_diag** – Non-zero, floating-point *Tensor* representing a diagonal matrix added to *scale*. May have shape *[B1, ..., Bb, k]*, *b* >= *0*, and characterizes *b*-batches of *k x k* diagonal matrices added to *scale*. When both *scale_identity_multiplier* and *scale_diag* are *None* then *scale* is the *Identity*.
- **scale_identity_multiplier** – Non-zero, floating-point *Tensor* representing a scaled-identity-matrix added to *scale*. May have shape *[B1, ..., Bb]*, *b* >= *0*, and characterizes *b*-batches of scaled *k x k* identity matrices added to *scale*. When both *scale_identity_multiplier* and *scale_diag* are *None* then *scale* is the *Identity*.
- **scale_perturb_factor** – Floating-point *Tensor* representing a rank-*r* perturbation added to *scale*. May have shape *[B1, ..., Bb, k, r]*, *b* >= *0*, and characterizes *b*-batches of rank-*r* updates to *scale*. When *None*, no rank-*r* update is added to *scale*.
- **scale_perturb_diag** – Floating-point *Tensor* representing a diagonal matrix inside the rank-*r* perturbation added to *scale*. May have shape *[B1, ..., Bb, r]*, *b* >= *0*, and characterizes *b*-batches of *r x r* diagonal matrices inside the perturbation added to *scale*. When *None*, an identity matrix is used inside the perturbation. Can only be specified if *scale_perturb_factor* is also specified.
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value “*NaN*” to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic’s batch members are undefined.
- **name** – Python *str* name prefixed to Ops created by this class.

Raises `ValueError` – if at most *scale_identity_multiplier* is specified.

`inferpy.models.`**`MultivariateNormalFullCovariance`**`(`**args*, ***kwargs*`)`

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for MultivariateNormalFullCovariance.

See MultivariateNormalFullCovariance for more details.

Returns RandomVariable.

Original Docstring for Distribution

Construct Multivariate Normal distribution on *R^k*.

The *batch_shape* is the broadcast shape between *loc* and *covariance_matrix* arguments.

The *event_shape* is given by last dimension of the matrix implied by *covariance_matrix*. The last dimension of *loc* (if provided) must broadcast with this.

A non-batch *covariance_matrix* matrix is a *k x k* symmetric positive definite matrix. In other words it is (real) symmetric with all eigenvalues strictly positive.

Additional leading dimensions (if any) will index batches.

Parameters

- **loc** – Floating-point *Tensor*. If this is set to *None*, *loc* is implicitly *0.* When specified, may have shape *[B1, . . . , Bb, k]* where *b >= 0* and *k* is the event size.
- **covariance_matrix** – Floating-point, symmetric positive definite *Tensor* of same *dtype* as *loc*. The strict upper triangle of *covariance_matrix* is ignored, so if *covariance_matrix* is not symmetric no error will be raised (unless *validate_args is True*). *covariance_matrix* has shape *[B1, . . . , Bb, k, k]* where *b >= 0* and *k* is the event size.
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined.
- **name** – Python *str* name prefixed to Ops created by this class.

Raises `ValueError` – if neither *loc* nor *covariance_matrix* are specified.

`inferpy.models.`**`MultivariateNormalLinearOperator`**`(`**args*, ***kwargs*`)`

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for MultivariateNormalLinearOperator.

See MultivariateNormalLinearOperator for more details.

Returns RandomVariable.

Original Docstring for Distribution

Construct Multivariate Normal distribution on *R^k*.

The *batch_shape* is the broadcast shape between *loc* and *scale* arguments.

The *event_shape* is given by last dimension of the matrix implied by *scale*. The last dimension of *loc* (if provided) must broadcast with this.

Recall that *covariance = scale @ scale.T*.

Additional leading dimensions (if any) will index batches.

Parameters

- **loc** – Floating-point *Tensor*. If this is set to *None*, *loc* is implicitly *0*. When specified, may have shape *[B1, ..., Bb, k]* where *b >= 0* and *k* is the event size.
- **scale** – Instance of *LinearOperator* with same *dtype* as *loc* and shape *[B1, ..., Bb, k, k]*.
- **validate_args** – Python *bool*, default *False*. Whether to validate input with asserts. If *validate_args* is *False*, and the inputs are invalid, correct behavior is not guaranteed.
- **allow_nan_stats** – Python *bool*, default *True*. If *False*, raise an exception if a statistic (e.g. mean/mode/etc...) is undefined for any batch member If *True*, batch members with valid parameters leading to undefined statistics will return NaN for this statistic.
- **name** – The name to give Ops created by the initializer.

Raises

- `ValueError` – if *scale* is unspecified.
- `TypeError` – if not *scale.dtype.is_floating*

`inferpy.models.`**`MultivariateNormalTriL`**`(*args, **kwargs)`

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for MultivariateNormalTriL.

See MultivariateNormalTriL for more details.

Returns RandomVariable.

Original Docstring for Distribution

Construct Multivariate Normal distribution on *R^k*.

The *batch_shape* is the broadcast shape between *loc* and *scale* arguments.

The *event_shape* is given by last dimension of the matrix implied by *scale*. The last dimension of *loc* (if provided) must broadcast with this.

Recall that *covariance* = *scale @ scale.T*. A (non-batch) *scale* matrix is:

`none scale = scale_tril `

where *scale_tril* is lower-triangular *k x k* matrix with non-zero diagonal, i.e., *tf.diag_part(scale_tril) != 0.*

Additional leading dimensions (if any) will index batches.

Parameters

- **loc** – Floating-point *Tensor*. If this is set to *None*, *loc* is implicitly *0.* When specified, may have shape *[B1, . . . , Bb, k]* where *b >= 0* and *k* is the event size.
- **scale_tril** – Floating-point, lower-triangular *Tensor* with non-zero diagonal elements. *scale_tril* has shape *[B1, . . . , Bb, k, k]* where *b >= 0* and *k* is the event size.
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined.
- **name** – Python *str* name prefixed to Ops created by this class.

Raises `ValueError` – if neither *loc* nor *scale_tril* are specified.

`inferpy.models.`**`NegativeBinomial`**`(*args, **kwargs)`

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for NegativeBinomial.

See NegativeBinomial for more details.

Returns RandomVariable.

Original Docstring for Distribution

Construct NegativeBinomial distributions.

Parameters

- **total_count** – Non-negative floating-point *Tensor* with shape broadcastable to *[B1,. . . , Bb]* with *b >= 0* and the same dtype as *probs* or *logits*. Defines this as a batch of *N1 x . . . x Nm* different Negative Binomial distributions. In practice, this represents the number of negative Bernoulli trials to stop at (the *total_count* of failures), but this is still a valid distribution when *total_count* is a non-integer.
- **logits** – Floating-point *Tensor* with shape broadcastable to *[B1, . . . , Bb]* where *b >= 0* indicates the number of batch dimensions. Each entry represents logits for the probability of success for independent Negative Binomial distributions and must be in the open interval *(-inf, inf)*. Only one of *logits* or *probs* should be specified.

- **probs** – Positive floating-point *Tensor* with shape broadcastable to *[B1, ..., Bb]* where *b >= 0* indicates the number of batch dimensions. Each entry represents the probability of success for independent Negative Binomial distributions and must be in the open interval *(0, 1)*. Only one of *logits* or *probs* should be specified.
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined.
- **name** – Python *str* name prefixed to Ops created by this class.

inferpy.models.**Normal**(**args*, ***kwargs*)

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for Normal.

See Normal for more details.

Returns RandomVariable.

Original Docstring for Distribution

Construct Normal distributions with mean and stddev *loc* and *scale*.

The parameters *loc* and *scale* must be shaped in a way that supports broadcasting (e.g. *loc + scale* is a valid operation).

Parameters

- **loc** – Floating point tensor; the means of the distribution(s).
- **scale** – Floating point tensor; the stddevs of the distribution(s). Must contain only positive values.
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined.
- **name** – Python *str* name prefixed to Ops created by this class.

Raises TypeError – if *loc* and *scale* have different *dtype*.

inferpy.models.**OneHotCategorical**(**args*, ***kwargs*)

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for OneHotCategorical.

See OneHotCategorical for more details.

Returns RandomVariable.

Original Docstring for Distribution

Initialize OneHotCategorical distributions using class log-probabilities.

Parameters

- **logits** – An N-D *Tensor*, *N >= 1*, representing the log probabilities of a set of Categorical distributions. The first *N - 1* dimensions index into a batch of independent distributions and the last dimension represents a vector of logits for each class. Only one of *logits* or *probs* should be passed in.
- **probs** – An N-D *Tensor*, *N >= 1*, representing the probabilities of a set of Categorical distributions. The first *N - 1* dimensions index into a batch of independent distributions and the last dimension represents a vector of probabilities for each class. Only one of *logits* or *probs* should be passed in.
- **dtype** – The type of the event samples (default: int32).
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined.
- **name** – Python *str* name prefixed to Ops created by this class.

inferpy.models.**Pareto**(**args*, ***kwargs*)

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for Pareto.

See Pareto for more details.

Returns RandomVariable.

Original Docstring for Distribution

Construct Pareto distribution with *concentration* and *scale*.

Parameters

- **concentration** – Floating point tensor. Must contain only positive values.
- **scale** – Floating point tensor, equivalent to *mode*. *scale* also restricts the domain of this distribution to be in *[scale, inf)*. Must contain only positive values. Default value: *1*.
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs. Default value: *False* (i.e. do not validate args).
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined. Default value: *True*.
- **name** – Python *str* name prefixed to Ops created by this class. Default value: 'Pareto'.

inferpy.models.**Poisson**(**args*, ***kwargs*)

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for Poisson.

See Poisson for more details.

Returns RandomVariable.

Original Docstring for Distribution

Initialize a batch of Poisson distributions.

Parameters

- **rate** – Floating point tensor, the rate parameter. *rate* must be positive. Must specify exactly one of *rate* and *log_rate*.
- **log_rate** – Floating point tensor, the log of the rate parameter. Must specify exactly one of *rate* and *log_rate*.
- **interpolate_nondiscrete** – Python *bool*. When *False*, *log_prob* returns *-inf* (and *prob* returns *0*) for non-integer inputs. When *True*, *log_prob* evaluates the continuous function *k * log_rate - lgamma(k+1) - rate*, which matches the Poisson pmf at integer arguments *k* (note that this function is not itself a normalized probability log-density). Default value: *True*.
- **validate_args** – Python *bool*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs. Default value: *False*.
- **allow_nan_stats** – Python *bool*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined. Default value: *True*.
- **name** – Python *str* name prefixed to Ops created by this class.

Raises

- ValueError – if none or both of *rate*, *log_rate* are specified.

- `TypeError` – if *rate* is not a float-type.
- `TypeError` – if *log_rate* is not a float-type.

`inferpy.models.`**`PoissonLogNormalQuadratureCompound`**`(*args, **kwargs)`

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for PoissonLogNormalQuadratureCompound.

See PoissonLogNormalQuadratureCompound for more details.

Returns RandomVariable.

Original Docstring for Distribution

Constructs the PoissonLogNormalQuadratureCompound‘.

Note: *probs* returned by (optional) *quadrature_fn* are presumed to be either a length-*quadrature_size* vector or a batch of vectors in 1-to-1 correspondence with the returned *grid*. (I.e., broadcasting is only partially supported.)

Parameters

- **`loc`** – *float*-like (batch of) scalar *Tensor*; the location parameter of the LogNormal prior.
- **`scale`** – *float*-like (batch of) scalar *Tensor*; the scale parameter of the LogNormal prior.
- **`quadrature_size`** – Python *int* scalar representing the number of quadrature points.
- **`quadrature_fn`** – Python callable taking *loc*, *scale*, *quadrature_size*, *validate_args* and returning *tuple(grid, probs)* representing the LogNormal grid and corresponding normalized weight. normalized) weight. Default value: *quadrature_scheme_lognormal_quantiles*.
- **`validate_args`** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **`allow_nan_stats`** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value “*NaN*” to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic’s batch members are undefined.
- **`name`** – Python *str* name prefixed to Ops created by this class.

Raises `TypeError` – if *quadrature_grid* and *quadrature_probs* have different base *dtype*.

`inferpy.models.`**`QuantizedDistribution`**`(*args, **kwargs)`

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for QuantizedDistribution.

See QuantizedDistribution for more details.

Returns RandomVariable.

Original Docstring for Distribution

Construct a Quantized Distribution representing *Y = ceiling(X)*.

Some properties are inherited from the distribution defining *X*. Example: *allow_nan_stats* is determined for this *QuantizedDistribution* by reading the *distribution*.

Parameters

- **distribution** – The base distribution class to transform. Typically an instance of *Distribution*.
- **low** – *Tensor* with same *dtype* as this distribution and shape able to be added to samples. Should be a whole number. Default *None*. If provided, base distribution's *prob* should be defined at *low*.
- **high** – *Tensor* with same *dtype* as this distribution and shape able to be added to samples. Should be a whole number. Default *None*. If provided, base distribution's *prob* should be defined at *high - 1*. *high* must be strictly greater than *low*.
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **name** – Python *str* name prefixed to Ops created by this class.

Raises

- `TypeError` – If *dist_cls* is not a subclass of *Distribution* or continuous.
- `NotImplementedError` – If the base distribution does not implement *cdf*.

`inferpy.models.`**`RelaxedBernoulli`**`(*args, **kwargs)`

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for RelaxedBernoulli.

See RelaxedBernoulli for more details.

Returns RandomVariable.

Original Docstring for Distribution

Construct RelaxedBernoulli distributions.

Parameters

- **temperature** – An 0-D *Tensor*, representing the temperature of a set of RelaxedBernoulli distributions. The temperature should be positive.

- **logits** – An N-D *Tensor* representing the log-odds of a positive event. Each entry in the *Tensor* parametrizes an independent RelaxedBernoulli distribution where the probability of an event is sigmoid(logits). Only one of *logits* or *probs* should be passed in.
- **probs** – An N-D *Tensor* representing the probability of a positive event. Each entry in the *Tensor* parameterizes an independent Bernoulli distribution. Only one of *logits* or *probs* should be passed in.
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined.
- **name** – Python *str* name prefixed to Ops created by this class.

Raises `ValueError` – If both *probs* and *logits* are passed, or if neither.

`inferpy.models.`**`ExpRelaxedOneHotCategorical`**`(`*args*, ***kwargs*`)`

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for ExpRelaxedOneHotCategorical.

See ExpRelaxedOneHotCategorical for more details.

Returns RandomVariable.

Original Docstring for Distribution

Initialize ExpRelaxedOneHotCategorical using class log-probabilities.

Parameters

- **temperature** – An 0-D *Tensor*, representing the temperature of a set of ExpRelaxedCategorical distributions. The temperature should be positive.
- **logits** – An N-D *Tensor*, *N >= 1*, representing the log probabilities of a set of ExpRelaxedCategorical distributions. The first *N - 1* dimensions index into a batch of independent distributions and the last dimension represents a vector of logits for each class. Only one of *logits* or *probs* should be passed in.
- **probs** – An N-D *Tensor*, *N >= 1*, representing the probabilities of a set of ExpRelaxedCategorical distributions. The first *N - 1* dimensions index into a batch of independent distributions and the last dimension represents a vector of probabilities for each class. Only one of *logits* or *probs* should be passed in.
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.

- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined.
- **name** – Python *str* name prefixed to Ops created by this class.

inferpy.models.**RelaxedOneHotCategorical**(**args*, ***kwargs*)

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for RelaxedOneHotCategorical.

See RelaxedOneHotCategorical for more details.

Returns RandomVariable.

Original Docstring for Distribution

Initialize RelaxedOneHotCategorical using class log-probabilities.

Parameters

- **temperature** – An 0-D *Tensor*, representing the temperature of a set of RelaxedOneHotCategorical distributions. The temperature should be positive.
- **logits** – An N-D *Tensor*, *N >= 1*, representing the log probabilities of a set of RelaxedOneHotCategorical distributions. The first *N - 1* dimensions index into a batch of independent distributions and the last dimension represents a vector of logits for each class. Only one of *logits* or *probs* should be passed in.
- **probs** – An N-D *Tensor*, *N >= 1*, representing the probabilities of a set of RelaxedOneHotCategorical distributions. The first *N - 1* dimensions index into a batch of independent distributions and the last dimension represents a vector of probabilities for each class. Only one of *logits* or *probs* should be passed in.
- **validate_args** – Unused in this distribution.
- **allow_nan_stats** – Python *bool*, default *True*. If *False*, raise an exception if a statistic (e.g. mean/mode/etc...) is undefined for any batch member. If *True*, batch members with valid parameters leading to undefined statistics will return NaN for this statistic.
- **name** – A name for this distribution (optional).

inferpy.models.**Sample**(**args*, ***kwargs*)

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for Sample.

See Sample for more details.

Returns RandomVariable.

Original Docstring for Distribution

Construct the *Sample* distribution.

Parameters

- **distribution** – The base distribution instance to transform. Typically an instance of *Distribution*.
- **sample_shape** – *int* scalar or vector *Tensor* representing the shape of a single sample.
- **validate_args** – Python *bool*. Whether to validate input with asserts. If *validate_args* is *False*, and the inputs are invalid, correct behavior is not guaranteed.
- **name** – The name for ops managed by the distribution. Default value: *None* (i.e., *'Sample' + distribution.name*).

inferpy.models.**SinhArcsinh**(**args*, ***kwargs*)

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for SinhArcsinh.

See SinhArcsinh for more details.

Returns RandomVariable.

Original Docstring for Distribution

Construct SinhArcsinh distribution on *(-inf, inf)*.

Arguments *(loc, scale, skewness, tailweight)* must have broadcastable shape (indexing batch dimensions). They must all have the same *dtype*.

Parameters

- **loc** – Floating-point *Tensor*.
- **scale** – *Tensor* of same *dtype* as *loc*.
- **skewness** – Skewness parameter. Default is *0.0* (no skew).
- **tailweight** – Tailweight parameter. Default is *1.0* (unchanged tailweight)
- **distribution** – *tf.Distribution*-like instance. Distribution that is transformed to produce this distribution. Default is *tfd.Normal(0., 1.)*. Must be a scalar-batch, scalar-event distribution. Typically *distribution.reparameterization_type = FULLY_REPARAMETERIZED* or it is a function of non-trainable parameters. WARNING: If you backprop through a *SinhArcsinh* sample and *distribution* is not *FULLY_REPARAMETERIZED* yet is a function of trainable variables, then the gradient will be incorrect!

- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined.
- **name** – Python *str* name prefixed to Ops created by this class.

inferpy.models.**StudentT**(**args*, ***kwargs*)

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for StudentT.

See StudentT for more details.

Returns RandomVariable.

Original Docstring for Distribution

Construct Student's t distributions.

The distributions have degree of freedom *df*, mean *loc*, and scale *scale*.

The parameters *df*, *loc*, and *scale* must be shaped in a way that supports broadcasting (e.g. *df* + *loc* + *scale* is a valid operation).

Parameters

- **df** – Floating-point *Tensor*. The degrees of freedom of the distribution(s). *df* must contain only positive values.
- **loc** – Floating-point *Tensor*. The mean(s) of the distribution(s).
- **scale** – Floating-point *Tensor*. The scaling factor(s) for the distribution(s). Note that *scale* is not technically the standard deviation of this distribution but has semantics more similar to standard deviation than variance.
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined.
- **name** – Python *str* name prefixed to Ops created by this class.

Raises `TypeError` – if loc and scale are different dtypes.

inferpy.models.**StudentTProcess**(**args*, ***kwargs*)

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for StudentTProcess.

See StudentTProcess for more details.

Returns RandomVariable.

Original Docstring for Distribution

Instantiate a StudentTProcess Distribution.

Parameters

- **df** – Positive Floating-point *Tensor* representing the degrees of freedom. Must be greater than 2.
- **kernel** – *PositiveSemidefiniteKernel*-like instance representing the TP's covariance function.
- **index_points** – *float Tensor* representing finite (batch of) vector(s) of points in the index set over which the TP is defined. Shape has the form *[b1, ..., bB, e, f1, ..., fF]* where *F* is the number of feature dimensions and must equal *kernel.feature_ndims* and *e* is the number (size) of index points in each batch. Ultimately this distribution corresponds to a *e*-dimensional multivariate Student's T. The batch shape must be broadcastable with *kernel.batch_shape* and any batch dims yielded by *mean_fn*.
- **mean_fn** – Python *callable* that acts on *index_points* to produce a (batch of) vector(s) of mean values at *index_points*. Takes a *Tensor* of shape *[b1, ..., bB, f1, ..., fF]* and returns a *Tensor* whose shape is broadcastable with *[b1, ..., bB]*. Default value: *None* implies constant zero function.
- **jitter** – *float* scalar *Tensor* added to the diagonal of the covariance matrix to ensure positive definiteness of the covariance matrix. Default value: *1e-6*.
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs. Default value: *False*.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined. Default value: *False*.
- **name** – Python *str* name prefixed to Ops created by this class. Default value: "StudentTProcess".

Raises `ValueError` – if *mean_fn* is not *None* and is not callable.

`inferpy.models.`**`ConditionalTransformedDistribution`**`(`**args*, ***kwargs*`)`

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for ConditionalTransformedDistribution.

See ConditionalTransformedDistribution for more details.

Returns RandomVariable.

Original Docstring for Distribution

Construct a Transformed Distribution.

Parameters

- **distribution** – The base distribution instance to transform. Typically an instance of *Distribution*.
- **bijector** – The object responsible for calculating the transformation. Typically an instance of *Bijector*.
- **batch_shape** – *integer* vector *Tensor* which overrides *distribution batch_shape*; valid only if *distribution.is_scalar_batch()*.
- **event_shape** – *integer* vector *Tensor* which overrides *distribution event_shape*; valid only if *distribution.is_scalar_event()*.
- **kwargs_split_fn** – Python *callable* which takes a kwargs *dict* and returns a tuple of kwargs *dict's for each of the 'distribution* and *bijector* parameters respectively. Default value: *_default_kwargs_split_fn* (i.e.,

  'lambda kwargs: (kwargs.get('distribution_kwargs', {}),
  kwargs.get('bijector_kwargs', {}))')
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **parameters** – Locals dict captured by subclass constructor, to be used for copy/slice re-instantiation operations.
- **name** – Python *str* name prefixed to Ops created by this class. Default: *bijector.name* + *distribution.name*.

inferpy.models.**TransformedDistribution**(**args*, ***kwargs*)

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for TransformedDistribution.

See TransformedDistribution for more details.

Returns RandomVariable.

Original Docstring for Distribution

Construct a Transformed Distribution.

Parameters

- **distribution** – The base distribution instance to transform. Typically an instance of *Distribution*.
- **bijector** – The object responsible for calculating the transformation. Typically an instance of *Bijector*.
- **batch_shape** – *integer* vector *Tensor* which overrides *distribution batch_shape*; valid only if *distribution.is_scalar_batch()*.
- **event_shape** – *integer* vector *Tensor* which overrides *distribution event_shape*; valid only if *distribution.is_scalar_event()*.
- **kwargs_split_fn** – Python *callable* which takes a kwargs *dict* and returns a tuple of kwargs *dict's for each of the 'distribution* and *bijector* parameters respectively. Default value: *_default_kwargs_split_fn* (i.e.,

  'lambda kwargs: (kwargs.get('distribution_kwargs', {}),
  kwargs.get('bijector_kwargs', {}))')
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **parameters** – Locals dict captured by subclass constructor, to be used for copy/slice re-instantiation operations.
- **name** – Python *str* name prefixed to Ops created by this class. Default: *bijector.name* + *distribution.name*.

inferpy.models.**Triangular**(**args*, ***kwargs*)

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for Triangular.

See Triangular for more details.

Returns RandomVariable.

Original Docstring for Distribution

Initialize a batch of Triangular distributions.

Parameters

- **low** – Floating point tensor, lower boundary of the output interval. Must have *low* < *high*. Default value: *0*.
- **high** – Floating point tensor, upper boundary of the output interval. Must have *low* < *high*. Default value: *1*.
- **peak** – Floating point tensor, mode of the output interval. Must have *low* <= *peak* and *peak* <= *high*. Default value: *0.5*.

- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs. Default value: *False*.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined. Default value: *True*.
- **name** – Python *str* name prefixed to Ops created by this class. Default value: *'Triangular'*.

Raises `InvalidArgumentError` – if *validate_args=True* and one of the following is True: * *low >= high*. * *peak > high*. * *low > peak*.

`inferpy.models.`**`TruncatedNormal`**`(*args, **kwargs)`

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for TruncatedNormal.

See TruncatedNormal for more details.

Returns RandomVariable.

Original Docstring for Distribution

Construct TruncatedNormal.

All parameters of the distribution will be broadcast to the same shape, so the resulting distribution will have a batch_shape of the broadcast shape of all parameters.

Parameters

- **loc** – Floating point tensor; the mean of the normal distribution(s) (note that the mean of the resulting distribution will be different since it is modified by the bounds).
- **scale** – Floating point tensor; the std deviation of the normal distribution(s).
- **low** – *float Tensor* representing lower bound of the distribution's support. Must be such that *low < high*.
- **high** – *float Tensor* representing upper bound of the distribution's support. Must be such that *low < high*.
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked at run-time.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined.
- **name** – Python *str* name prefixed to Ops created by this class.

`inferpy.models.`**`Uniform`**`(*args, **kwargs)`

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for Uniform.

See Uniform for more details.

Returns RandomVariable.

Original Docstring for Distribution

Initialize a batch of Uniform distributions.

Parameters

- **low** – Floating point tensor, lower boundary of the output interval. Must have *low* < *high*.
- **high** – Floating point tensor, upper boundary of the output interval. Must have *low* < *high*.
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined.
- **name** – Python *str* name prefixed to Ops created by this class.

Raises `InvalidArgumentError` – if *low* >= *high* and *validate_args*=*False*.

`inferpy.models.`**`VariationalGaussianProcess`**`(`**args*, ***kwargs*`)`

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for VariationalGaussianProcess.

See VariationalGaussianProcess for more details.

Returns RandomVariable.

Original Docstring for Distribution

Instantiate a VariationalGaussianProcess Distribution.

Parameters

- **kernel** – *PositiveSemidefiniteKernel*-like instance representing the GP's covariance function.
- **index_points** – *float Tensor* representing finite (batch of) vector(s) of points in the index set over which the VGP is defined. Shape has the form *[b1, ..., bB, e1, f1, ..., fF]* where *F* is the number of feature dimensions and must equal *kernel.feature_ndims* and *e1* is the

number (size) of index points in each batch (we denote it *e1* to distinguish it from the numer of inducing index points, denoted *e2* below). Ultimately the VariationalGaussianProcess distribution corresponds to an *e1*-dimensional multivariate normal. The batch shape must be broadcastable with *kernel.batch_shape*, the batch shape of *inducing_index_points*, and any batch dims yielded by *mean_fn*.

- **inducing_index_points** – *float Tensor* of locations of inducing points in the index set. Shape has the form *[b1, ..., bB, e2, f1, ..., fF]*, just like *index_points*. The batch shape components needn't be identical to those of *index_points*, but must be broadcast compatible with them.
- **variational_inducing_observations_loc** – *float Tensor*; the mean of the (full-rank Gaussian) variational posterior over function values at the inducing points, conditional on observed data. Shape has the form *[b1, ..., bB, e2]*, where *b1, ..., bB* is broadcast compatible with other parameters' batch shapes, and *e2* is the number of inducing points.
- **variational_inducing_observations_scale** – *float Tensor*; the scale matrix of the (full-rank Gaussian) variational posterior over function values at the inducing points, conditional on observed data. Shape has the form *[b1, ..., bB, e2, e2]*, where *b1, ..., bB* is broadcast compatible with other parameters and *e2* is the number of inducing points.
- **mean_fn** – Python *callable* that acts on index points to produce a (batch of) vector(s) of mean values at those index points. Takes a *Tensor* of shape *[b1, ..., bB, f1, ..., fF]* and returns a *Tensor* whose shape is (broadcastable with) *[b1, ..., bB]*. Default value: *None* implies constant zero function.
- **observation_noise_variance** – *float Tensor* representing the variance of the noise in the Normal likelihood distribution of the model. May be batched, in which case the batch shape must be broadcastable with the shapes of all other batched parameters (*kernel.batch_shape*, *index_points*, etc.). Default value: *0.*
- **predictive_noise_variance** – *float Tensor* representing additional variance in the posterior predictive model. If *None*, we simply re-use *observation_noise_variance* for the posterior predictive noise. If set explicitly, however, we use the given value. This allows us, for example, to omit predictive noise variance (by setting this to zero) to obtain noiseless posterior predictions of function values, conditioned on noisy observations.
- **jitter** – *float* scalar *Tensor* added to the diagonal of the covariance matrix to ensure positive definiteness of the covariance matrix. Default value: *1e-6*.
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs. Default value: *False*.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined. Default value: *False*.
- **name** – Python *str* name prefixed to Ops created by this class. Default value: "VariationalGaussianProcess".

Raises `ValueError` – if *mean_fn* is not *None* and is not callable.

`inferpy.models.`**`VectorDiffeomixture`**`(*args, **kwargs)`

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.

- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for VectorDiffeomixture.

See VectorDiffeomixture for more details.

Returns RandomVariable.

Original Docstring for Distribution

Constructs the VectorDiffeomixture on *R^d*.

The vector diffeomixture (VDM) approximates the compound distribution

```
`none p(x) = int p(x | z) p(z) dz, where z is in the K-simplex, and p(x | z) := p(x | loc=sum_k z[k] loc[k], scale=sum_k z[k] scale[k]) `
```

Parameters

- **mix_loc** – *float*-like *Tensor* with shape *[b1, ..., bB, K-1]*. In terms of samples, larger *mix_loc[..., k] ==> Z* is more likely to put more weight on its *kth* component.
- **temperature** – *float*-like *Tensor*. Broadcastable with *mix_loc*. In terms of samples, smaller *temperature* means one component is more likely to dominate. I.e., smaller *temperature* makes the VDM look more like a standard mixture of *K* components.
- **distribution** – *tfp.distributions.Distribution*-like instance. Distribution from which *d* iid samples are used as input to the selected affine transformation. Must be a scalar-batch, scalar-event distribution. Typically *distribution.reparameterization_type = FULLY_REPARAMETERIZED* or it is a function of non-trainable parameters. WARNING: If you backprop through a VectorDiffeomixture sample and the *distribution* is not *FULLY_REPARAMETERIZED* yet is a function of trainable variables, then the gradient will be incorrect!
- **loc** – Length-*K* list of *float*-type *Tensor's. The 'k*-th element represents the *shift* used for the *k*-th affine transformation. If the *k*-th item is *None*, *loc* is implicitly *0*. When specified, must have shape *[B1, ..., Bb, d]* where *b >= 0* and *d* is the event size.
- **scale** – Length-*K* list of *LinearOperator's. Each should be positive-definite and operate on a 'd*-dimensional vector space. The *k*-th element represents the *scale* used for the *k*-th affine transformation. *LinearOperator's must have shape '[B1, ..., Bb, d, d]*, *b >= 0*, i.e., characterizes *b*-batches of *d x d* matrices
- **quadrature_size** – Python *int* scalar representing number of quadrature points. Larger *quadrature_size* means *q_N(x)* better approximates *p(x)*.
- **quadrature_fn** – Python callable taking *normal_loc*, *normal_scale*, *quadrature_size*, *validate_args* and returning *tuple(grid, probs)* representing the SoftmaxNormal grid and corresponding normalized weight. normalized) weight. Default value: *quadrature_scheme_softmaxnormal_quantiles*.
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined.
- **name** – Python *str* name prefixed to Ops created by this class.

Raises

- `ValueError` – if *not scale or len(scale) < 2*.
- `ValueError` – if *len(loc) != len(scale)*
- `ValueError` – if *quadrature_grid_and_probs is not None* and *len(quadrature_grid_and_probs[0]) != len(quadrature_grid_and_probs[1])*
- `ValueError` – if *validate_args* and any not scale.is_positive_definite.
- `TypeError` – if any scale.dtype != scale[0].dtype.
- `TypeError` – if any loc.dtype != scale[0].dtype.
- `NotImplementedError` – if *len(scale) != 2*.
- `ValueError` – if *not distribution.is_scalar_batch*.
- `ValueError` – if *not distribution.is_scalar_event*.

`inferpy.models.`**`VectorExponentialDiag`**`(*args, **kwargs)`

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for VectorExponentialDiag.

See VectorExponentialDiag for more details.

Returns RandomVariable.

Original Docstring for Distribution

Construct Vector Exponential distribution supported on a subset of *R^k*.

The *batch_shape* is the broadcast shape between *loc* and *scale* arguments.

The *event_shape* is given by last dimension of the matrix implied by *scale*. The last dimension of *loc* (if provided) must broadcast with this.

Recall that *covariance = scale @ scale.T*.

`` `none scale = diag(scale_diag + scale_identity_multiplier * ones(k)) ` ``

where:

- *scale_diag.shape = [k]*, and,
- *scale_identity_multiplier.shape = []*.

Additional leading dimensions (if any) will index batches.

If both *scale_diag* and *scale_identity_multiplier* are *None*, then *scale* is the Identity matrix.

Parameters

- **`loc`** – Floating-point *Tensor*. If this is set to *None*, *loc* is implicitly *0*. When specified, may have shape *[B1, ..., Bb, k]* where *b >= 0* and *k* is the event size.

- **scale_diag** – Non-zero, floating-point *Tensor* representing a diagonal matrix added to *scale*. May have shape *[B1, ..., Bb, k]*, *b >= 0*, and characterizes *b*-batches of *k x k* diagonal matrices added to *scale*. When both *scale_identity_multiplier* and *scale_diag* are *None* then *scale* is the *Identity*.
- **scale_identity_multiplier** – Non-zero, floating-point *Tensor* representing a scaled-identity-matrix added to *scale*. May have shape *[B1, ..., Bb]*, *b >= 0*, and characterizes *b*-batches of scaled *k x k* identity matrices added to *scale*. When both *scale_identity_multiplier* and *scale_diag* are *None* then *scale* is the *Identity*.
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined.
- **name** – Python *str* name prefixed to Ops created by this class.

Raises `ValueError` – if at most *scale_identity_multiplier* is specified.

`inferpy.models.`**`VectorLaplaceDiag`**`(*args, **kwargs)`

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for VectorLaplaceDiag.

See VectorLaplaceDiag for more details.

Returns RandomVariable.

Original Docstring for Distribution

Construct Vector Laplace distribution on *R^k*.

The *batch_shape* is the broadcast shape between *loc* and *scale* arguments.

The *event_shape* is given by last dimension of the matrix implied by *scale*. The last dimension of *loc* (if provided) must broadcast with this.

Recall that *covariance = 2 * scale @ scale.T*.

`` `none scale = diag(scale_diag + scale_identity_multiplier * ones(k)) ` ``

where:

- *scale_diag.shape = [k]*, and,
- *scale_identity_multiplier.shape = []*.

Additional leading dimensions (if any) will index batches.

If both *scale_diag* and *scale_identity_multiplier* are *None*, then *scale* is the Identity matrix.

Parameters

- **loc** – Floating-point *Tensor*. If this is set to *None*, *loc* is implicitly *0.* When specified, may have shape *[B1, ..., Bb, k]* where *b >= 0* and *k* is the event size.
- **scale_diag** – Non-zero, floating-point *Tensor* representing a diagonal matrix added to *scale*. May have shape *[B1, ..., Bb, k]*, *b >= 0*, and characterizes *b*-batches of *k x k* diagonal matrices added to *scale*. When both *scale_identity_multiplier* and *scale_diag* are *None* then *scale* is the *Identity*.
- **scale_identity_multiplier** – Non-zero, floating-point *Tensor* representing a scaled-identity-matrix added to *scale*. May have shape *[B1, ..., Bb]*, *b >= 0*, and characterizes *b*-batches of scaled *k x k* identity matrices added to *scale*. When both *scale_identity_multiplier* and *scale_diag* are *None* then *scale* is the *Identity*.
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined.
- **name** – Python *str* name prefixed to Ops created by this class.

Raises `ValueError` – if at most *scale_identity_multiplier* is specified.

`inferpy.models.`**`VectorSinhArcsinhDiag`**`(`**args*, ***kwargs*`)`

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for VectorSinhArcsinhDiag.

See VectorSinhArcsinhDiag for more details.

Returns RandomVariable.

Original Docstring for Distribution

Construct VectorSinhArcsinhDiag distribution on *R^k*.

The arguments *scale_diag* and *scale_identity_multiplier* combine to define the diagonal *scale* referred to in this class docstring:

```
`none scale = diag(scale_diag + scale_identity_multiplier * ones(k)) `
```

The *batch_shape* is the broadcast shape between *loc* and *scale* arguments.

The *event_shape* is given by last dimension of the matrix implied by *scale*. The last dimension of *loc* (if provided) must broadcast with this

Additional leading dimensions (if any) will index batches.

Parameters

- **loc** – Floating-point *Tensor*. If this is set to *None*, *loc* is implicitly *0.* When specified, may have shape *[B1, ..., Bb, k]* where *b >= 0* and *k* is the event size.

- **scale_diag** – Non-zero, floating-point *Tensor* representing a diagonal matrix added to *scale*. May have shape *[B1, ..., Bb, k]*, *b >= 0*, and characterizes *b*-batches of *k x k* diagonal matrices added to *scale*. When both *scale_identity_multiplier* and *scale_diag* are *None* then *scale* is the *Identity*.
- **scale_identity_multiplier** – Non-zero, floating-point *Tensor* representing a scale-identity-matrix added to *scale*. May have shape *[B1, ..., Bb]*, *b >= 0*, and characterizes *b*-batches of scale *k x k* identity matrices added to *scale*. When both *scale_identity_multiplier* and *scale_diag* are *None* then *scale* is the *Identity*.
- **skewness** – Skewness parameter. floating-point *Tensor* with shape broadcastable with *event_shape*.
- **tailweight** – Tailweight parameter. floating-point *Tensor* with shape broadcastable with *event_shape*.
- **distribution** – *tf.Distribution*-like instance. Distribution from which *k* iid samples are used as input to transformation *F*. Default is *tfd.Normal(loc=0., scale=1.)*. Must be a scalar-batch, scalar-event distribution. Typically *distribution.reparameterization_type = FULLY_REPARAMETERIZED* or it is a function of non-trainable parameters. WARNING: If you backprop through a VectorSinhArcsinhDiag sample and *distribution* is not *FULLY_REPARAMETERIZED* yet is a function of trainable variables, then the gradient will be incorrect!
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined.
- **name** – Python *str* name prefixed to Ops created by this class.

Raises `ValueError` – if at most *scale_identity_multiplier* is specified.

`inferpy.models.`**`VonMises`**`(*args, **kwargs)`

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for VonMises.

See VonMises for more details.

Returns RandomVariable.

Original Docstring for Distribution

Construct von Mises distributions with given location and concentration.

The parameters *loc* and *concentration* must be shaped in a way that supports broadcasting (e.g. *loc + concentration* is a valid operation).

Parameters

- **loc** – Floating point tensor, the circular means of the distribution(s).
- **concentration** – Floating point tensor, the level of concentration of the distribution(s) around *loc*. Must take non-negative values. *concentration = 0* defines a Uniform distribution, while *concentration = +inf* indicates a Deterministic distribution at *loc*.
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined.
- **name** – Python *str* name prefixed to Ops created by this class.

Raises `TypeError` – if loc and concentration are different dtypes.

`inferpy.models.`**`VonMisesFisher`**(**args*, ***kwargs*)

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for VonMisesFisher.

See VonMisesFisher for more details.

Returns RandomVariable.

Original Docstring for Distribution

Creates a new *VonMisesFisher* instance.

Parameters

- **mean_direction** – Floating-point *Tensor* with shape [B1, ... Bn, D]. A unit vector indicating the mode of the distribution, or the unit-normalized direction of the mean. (This is *not* in general the mean of the distribution; the mean is not generally in the support of the distribution.) NOTE: *D* is currently restricted to <= 5.
- **concentration** – Floating-point *Tensor* having batch shape [B1, ... Bn] broadcastable with *mean_direction*. The level of concentration of samples around the *mean_direction*. *concentration=0* indicates a uniform distribution over the unit hypersphere, and *concentration=+inf* indicates a *Deterministic* distribution (delta function) at *mean_direction*.
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined.
- **name** – Python *str* name prefixed to Ops created by this class.

Raises `ValueError` – For known-bad arguments, i.e. unsupported event dimension.

inferpy.models.**Wishart**(**args*, ***kwargs*)

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.
- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for Wishart.

See Wishart for more details.

Returns RandomVariable.

Original Docstring for Distribution

Construct Wishart distributions.

Parameters

- **df** – *float* or *double Tensor*. Degrees of freedom, must be greater than or equal to dimension of the scale matrix.
- **scale** – *float* or *double Tensor*. The symmetric positive definite scale matrix of the distribution. Exactly one of *scale* and 'scale_tril' must be passed.
- **scale_tril** – *float* or *double Tensor*. The Cholesky factorization of the symmetric positive definite scale matrix of the distribution. Exactly one of *scale* and 'scale_tril' must be passed.
- **input_output_cholesky** – Python *bool*. If *True*, functions whose input or output have the semantics of samples assume inputs are in Cholesky form and return outputs in Cholesky form. In particular, if this flag is *True*, input to *log_prob* is presumed of Cholesky form and output from *sample*, *mean*, and *mode* are of Cholesky form. Setting this argument to *True* is purely a computational optimization and does not change the underlying distribution; for instance, *mean* returns the Cholesky of the mean, not the mean of Cholesky factors. The *variance* and *stddev* methods are unaffected by this flag. Default value: *False* (i.e., input/output does not have Cholesky semantics).
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined.
- **name** – Python *str* name prefixed to Ops created by this class.

Raises `ValueError` – if zero or both of 'scale' and 'scale_tril' are passed in.

inferpy.models.**Zipf**(**args*, ***kwargs*)

Class for random variables. It encapsulates the Random Variable from edward2, and additional properties.

- It creates a variable generator. It must be a function without parameters, that creates a new Random Variable from edward2. It is used to define edward2 models as functions. Also, it is useful to define models using the intercept function.

- The first time the var property is used, it creates a var using the variable generator.

Random Variable information:

Create a random variable for Zipf.

See Zipf for more details.

Returns RandomVariable.

Original Docstring for Distribution

Initialize a batch of Zipf distributions.

Parameters

- **power** – *Float* like *Tensor* representing the power parameter. Must be strictly greater than *1*.
- **dtype** – The *dtype* of *Tensor* returned by *sample*. Default value: *tf.int32*.
- **interpolate_nondiscrete** – Python *bool*. When *False*, *log_prob* returns *-inf* (and *prob* returns *0*) for non-integer inputs. When *True*, *log_prob* evaluates the continuous function *-power log(k) - log(zeta(power))* , which matches the Zipf pmf at integer arguments *k* (note that this function is not itself a normalized probability log-density). Default value: *True*.
- **sample_maximum_iterations** – Maximum number of iterations of allowable iterations in *sample*. When *validate_args=True*, samples which fail to reach convergence (subject to this cap) are masked out with *self.dtype.min* or *nan* depending on *self.dtype.is_integer*. Default value: *100*.
- **validate_args** – Python *bool*, default *False*. When *True* distribution parameters are checked for validity despite possibly degrading runtime performance. When *False* invalid inputs may silently render incorrect outputs. Default value: *False*.
- **allow_nan_stats** – Python *bool*, default *True*. When *True*, statistics (e.g., mean, mode, variance) use the value "*NaN*" to indicate the result is undefined. When *False*, an exception is raised if one or more of the statistic's batch members are undefined. Default value: *False*.
- **name** – Python *str* name prefixed to Ops created by this class. Default value: *'Zipf'*.

Raises `TypeError` – if *power* is not *float* like.

8.1.5 inferpy.queries package

Submodules

inferpy.queries.query module

class `inferpy.queries.query.`**Query**(*variables*, *target_names=None*, *data={}*)

Bases: `object`

log_prob()

Computes the log probabilities of a (set of) sample(s)

parameters(*names=None*)

Return the parameters of the Random Variables of the model. If *names* is None, then return all the parameters of all the Random Variables. If *names* is a list, then return the parameters specified in the list (if exists) for all the Random Variables. If *names* is a dict, then return all the parameters specified (value) for each Random Variable (key).

NOTE: If tf_run=True, but any of the returned parameters is not a Tensor *and therefore cannot be evaluated)
this returns a not evaluated dict (because the evaluation will raise an Exception)

Parameters **names** – A list, a dict or None. Specify the parameters for the Random Variables to be obtained.

Returns A dict, where the keys are the names of the Random Variables and the values a dict of parameters (name-value)

sample(*size=1*)
Generates a sample for eache variable in the model

sum_log_prob()
Computes the sum of the log probabilities (evaluated) of a (set of) sample(s)

inferpy.queries.query.**flatten_result**(*f*)

Module contents

class inferpy.queries.**Query**(*variables*, *target_names=None*, *data={}*)
Bases: object

log_prob()
Computes the log probabilities of a (set of) sample(s)

parameters(*names=None*)
Return the parameters of the Random Variables of the model. If *names* is None, then return all the parameters of all the Random Variables. If *names* is a list, then return the parameters specified in the list (if exists) for all the Random Variables. If *names* is a dict, then return all the parameters specified (value) for each Random Variable (key).

NOTE: If tf_run=True, but any of the returned parameters is not a Tensor *and therefore cannot be evaluated)
this returns a not evaluated dict (because the evaluation will raise an Exception)

Parameters **names** – A list, a dict or None. Specify the parameters for the Random Variables to be obtained.

Returns A dict, where the keys are the names of the Random Variables and the values a dict of parameters (name-value)

sample(*size=1*)
Generates a sample for eache variable in the model

sum_log_prob()
Computes the sum of the log probabilities (evaluated) of a (set of) sample(s)

8.1.6 inferpy.util package

Submodules

inferpy.util.interceptor module

inferpy.util.interceptor.**set_values**(***model_kwargs*)
Creates a value-setting interceptor. Usable as a parameter of the ed2.interceptor.

Model_kwargs The name of each argument must be the name of a random variable to intercept, and the value is the element which intercepts the value of the random variable.

Returns The random variable with the intercepted value

inferpy.util.interceptor.**set_values_condition**(*var_condition*, *var_value*)
Creates a value-setting interceptor. Usable as a parameter of the ed2.interceptor.

Var_condition (tf.Variable)*tf.Variable*) The boolean tf.Variable, used to intercept the value property with *value_var* or the variable value property itself

Var_value (tf.Variable)*tf.Variable*) The tf.Variable used to intercept the value property when *var_condition* is True

Returns The random variable with the intercepted value

inferpy.util.iterables module

inferpy.util.iterables.**get_plate_size**(*variables*, *sample_dict*)

inferpy.util.iterables.**get_shape**(*x*)
Get the shape of an element x. If it is an element with a shape attribute, return it. If it is a list with more than one element, compute the shape by checking the len, and the shape of internal elements. In that case, the shape must be consistent. Finally, in other case return () as shape.

Parameters **x** – The element to compute its shape

:raises : class *ValueError*: list shape not consistent :returns: A tuple with the shape of *x*

inferpy.util.name module

inferpy.util.name.**generate**(*prefix*)
This function is used to generate names based on an incremental counter (global variable in this module) dependent on the prefix (staring from 0 index)

Prefix (str)*str*) The begining of the random generated name

Returns The generated random name

inferpy.util.runtime module

Module focused on evaluating tensors to makes the usage easier, forgetting about tensors and sessions

inferpy.util.runtime.**runner_scope**()

inferpy.util.runtime.**set_tf_run**(*enable*)

inferpy.util.runtime.**tf_run_allowed**(*f*)
A function might return a tensor or not. In order to decide if the result of this function needs to be evaluated in a tf session or not, use the tf_run extra parameter or the tf_run_default value. If True, and this function is in the first level of execution depth, use a tf Session to evaluate the tensor or other evaluable object (like dicts)

inferpy.util.runtime.**tf_run_ignored**(*f*)
A function might call other functions decorated with tf_run_allowed. This decorator is used to avoid that such functions are evaluated.

inferpy.util.runtime.**try_run**(*obj*)

inferpy.util.session module

inferpy.util.session.**clear_session**()

inferpy.util.session.**get_session**()

inferpy.util.session.**new_session**(*gpu_memory_fraction=0.0*)

inferpy.util.session.**set_session**(*session*)

inferpy.util.session.**swap_session**(*new_session*)

inferpy.util.tf_graph module

inferpy.util.tf_graph.**get_empty_graph**()

inferpy.util.tf_graph.**get_graph**(*varnames*)

Module contents

Package with modules defining functions, classes and variables which are useful for the main functionality provided by inferpy

inferpy.util.**floatx**()
Returns the default float type, as a string. (e.g. 'float16', 'float32', 'float64'). # Returns

String, the current default float type.

Example ```python

```
>>> inf.floatx()
'float32'
```

```

inferpy.util.**set_floatx**(*floatx*)
Sets the default float type. # Arguments

floatx: String, 'float16', 'float32', or 'float64'.

# Example ```python

```
>>> from keras import backend as K
>>> inf.floatx()
'float32'
>>> inf.set_floatx('float16')
>>> inf..floatx()
'float16'
```

```

inferpy.util.**set_tf_run**(*enable*)

inferpy.util.**tf_run_allowed**(*f*)
A function might return a tensor or not. In order to decide if the result of this function needs to be evaluated in a tf session or not, use the tf_run extra parameter or the tf_run_default value. If True, and this function is in the first level of execution depth, use a tf Session to evaluate the tensor or other evaluable object (like dicts)

inferpy.util.**tf_run_ignored**(*f*)
A function might call other functions decorated with tf_run_allowed. This decorator is used to avoid that such functions are evaluated.

inferpy.util.**get_session**()

inferpy.util.**set_session**(*session*)

inferpy.util.**clear_session**()

8.2 Module contents

CHAPTER

NINE

CONTACT AND SUPPORT

If you have any question about the toolbox or if you want to collaborate in the project, please do not hesitate to contact us. You can do it through the following email address: inferpy.api@gmail.com

For more technical questions, please use Github issues.

PYTHON MODULE INDEX

c

d

i

m

q

u

INDEX

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

Z