
Pixyz Documentation

masa

Dec 12, 2018

Package Reference

1	pixyz.distributions (Distribution API)	3
2	pixyz.losses (Loss API)	13
3	pixyz.models (Model API)	19
4	pixyz.utils	21
5	Indices and tables	23
	Python Module Index	25

Pixyz is a library for developing deep generative models in a more concise, intuitive and extendable way!

pixyz.distributions (Distribution API)

1.1 Distribution

```
class pixyz.distributions.distributions.Distribution (cond_var=[], var=['x'],  
                                                name='p', dim=1)
```

Bases: torch.nn.modules.module.Module

Distribution class. In pixyz, all distributions are required to inherit this class.

var [list] Variables of this distribution.

cond_var [list] Conditional variables of this distribution. In case that `cond_var` is not empty, we must set the corresponding inputs in order to sample variables or estimate the log likelihood.

dim [int] Number of dimensions of this distribution. This might be ignored depending on the shape which is set in the `sample` method and on its parent distribution. Moreover, this is not consider when this class is inherited by DNNs. This is set to 1 by default.

name [str] Name of this distribution. This name is displayed in `prob_text` and `prob_factorized_text`. This is set to “p” by default.

distribution_name

name

var

cond_var

input_var

Normally, `input_var` has same values as `cond_var`.

prob_text

prob_factorized_text

get_params (*params_dict*)

This method aims to get parameters of this distributions from constant parameters set in initialization and outputs of DNNs.

params_dict [dict] Input parameters.

output_dict [dict] Output parameters

```
>>> print(dist_1.prob_text, dist_1.distribution_name)
p(x) Normal
>>> dist_1.get_params()
{"loc": 0, "scale": 1}
>>> print(dist_2.prob_text, dist_2.distribution_name)
p(x|z) Normal
>>> dist_1.get_params({"z": 1})
{"loc": 0, "scale": 1}
```

sample ($x=\{\}$, $shape=None$, $batch_size=1$, $return_all=True$, $reparam=False$)

Sample variables of this distribution. If *cond_var* is not empty, we should set inputs as a dictionary format.

x [torch.Tensor, list, or dict] Input variables.

shape [tuple] Shape of samples. If set, *batch_size* and *dim* are ignored.

batch_size [int] Batch size of samples. This is set to 1 by default.

return_all [bool] Choose whether the output contains input variables.

reparam [bool] Choose whether we sample variables with reparameterized trick.

output [dict] Samples of this distribution.

log_likelihood (x_dict)

Estimate the log likelihood of this distribution from inputs formatted by a dictionary.

x_dict [dict] Input samples.

log_like [torch.Tensor] Log-likelihood.

forward ($*args$, $**kwargs$)

When this class is inherited by DNNs, it is also intended that this method is overridden.

sample_mean (x)

replace_var ($**replace_dict$)

marginalize_var ($marginalize_list$)

1.2 Exponential families

1.2.1 Normal

class pixyz.distributions.**Normal** ($cond_var=[]$, $var=['x']$, $name='p'$, $dim=1$, $**kwargs$)

Bases: pixyz.distributions.distributions.DistributionBase

distribution_name

sample_mean (x)

1.2.2 Bernoulli

```
class pixyz.distributions.Bernoulli (cond_var=[], var=['x'], name='p', dim=1, **kwargs)
    Bases: pixyz.distributions.distributions.DistributionBase

    distribution_name

    sample_mean (x)
```

1.2.3 RelaxedBernoulli

```
class pixyz.distributions.RelaxedBernoulli (temperature, cond_var=[], var=['x'],
                                           name='p', dim=1, **kwargs)
    Bases: pixyz.distributions.distributions.DistributionBase

    distribution_name

    log_likelihood (x)
        Estimate the log likelihood of this distribution from inputs formatted by a dictionary.

        x_dict [dict] Input samples.

        log_like [torch.Tensor] Log-likelihood.

    sample_mean (x)
```

1.2.4 FactorizedBernoulli

```
class pixyz.distributions.FactorizedBernoulli (cond_var=[], var=['x'], name='p',
                                               dim=1, **kwargs)
    Bases: pixyz.distributions.exponential_distributions.Bernoulli

    Generative Models of Visually Grounded Imagination

    distribution_name
```

1.2.5 Categorical

```
class pixyz.distributions.Categorical (one_hot=True, cond_var=[], var=['x'], name='p',
                                       dim=1, **kwargs)
    Bases: pixyz.distributions.distributions.DistributionBase

    distribution_name

    sample_mean (x)
```

1.2.6 RelaxedCategorical

```
class pixyz.distributions.RelaxedCategorical (temperature, cond_var=[], var=['x'],
                                              name='p', dim=1, **kwargs)
    Bases: pixyz.distributions.distributions.DistributionBase

    distribution_name

    log_likelihood (x)
        Estimate the log likelihood of this distribution from inputs formatted by a dictionary.
```

x_dict [dict] Input samples.

log_like [torch.Tensor] Log-likelihood.

sample_mean (*x*)

1.3 Special distributions

1.3.1 NormalPoE

class `pixyz.distributions.NormalPoE` (*prior*, *dists=[]*, ***kwargs*)

Bases: `torch.nn.modules.module.Module`

$$p(z|x, y) \propto p(z)p(z|x)p(z|y)$$

dists [list] Other distributions.

prior [Distribution] Prior distribution.

```
>>> poe = NormalPoE(c, [a, b])
```

get_params (*params*, ***kwargs*)

experts (*loc*, *scale*, *eps=1e-08*)

sample (*x=None*, *return_all=True*, ***kwargs*)

log_likelihood (*x*)

sample_mean (*x*, ***kwargs*)

1.3.2 Deterministic

class `pixyz.distributions.Deterministic` (***kwargs*)

Bases: `pixyz.distributions.distributions.Distribution`

Deterministic distribution (or degeneration distribution)

distribution_name

sample (*x={}*, *return_all=True*, ***kwargs*)

Sample variables of this distribution. If *cond_var* is not empty, we should set inputs as a dictionary format.

x [torch.Tensor, list, or dict] Input variables.

shape [tuple] Shape of samples. If set, *batch_size* and *dim* are ignored.

batch_size [int] Batch size of samples. This is set to 1 by default.

return_all [bool] Choose whether the output contains input variables.

reparam [bool] Choose whether we sample variables with reparameterized trick.

output [dict] Samples of this distribution.

1.3.3 DataDistribution

class `pixyz.distributions.DataDistribution` (*var*, *name='p_data'*)

Bases: `pixyz.distributions.distributions.Distribution`

Data distribution. TODO: Fix this behavior if multiplied with other distributions

distribution_name

sample (*x={}*, ***kwargs*)

Sample variables of this distribution. If *cond_var* is not empty, we should set inputs as a dictionary format.

x [torch.Tensor, list, or dict] Input variables.

shape [tuple] Shape of samples. If set, *batch_size* and *dim* are ignored.

batch_size [int] Batch size of samples. This is set to 1 by default.

return_all [bool] Choose whether the output contains input variables.

reparam [bool] Choose whether we sample variables with reparameterized trick.

output [dict] Samples of this distribution.

input_var

In `DataDistribution`, *input_var* is same as *var*.

1.3.4 CustomLikelihoodDistribution

class `pixyz.distributions.CustomLikelihoodDistribution` (*var=['x']*, *likelihood=None*, ***kwargs*)

Bases: `pixyz.distributions.distributions.Distribution`

input_var

In `CustomLikelihoodDistribution`, *input_var* is same as *var*.

distribution_name

log_likelihood (*x_dict*)

Estimate the log likelihood of this distribution from inputs formatted by a dictionary.

x_dict [dict] Input samples.

log_like [torch.Tensor] Log-likelihood.

1.4 Flow-based

1.4.1 PlanarFlow

class `pixyz.distributions.PlanarFlow` (*prior*, *dim*, *num_layers=1*, *var=[]*, ***kwargs*)

Bases: `pixyz.distributions.flows.Flow`

1.4.2 RealNVP

class `pixyz.distributions.RealNVP` (*prior, dim, num_multiscale_layers=2, var=[], image=False, name='p', **kwargs*)

Bases: `pixyz.distributions.distributions.Distribution`

prob_text

forward (*x, inverse=False, jacobian=False*)

When this class is inherited by DNNs, it is also intended that this method is overridden.

sample (*x={}, only_flow=False, return_all=True, **kwargs*)

Sample variables of this distribution. If *cond_var* is not empty, we should set inputs as a dictionary format.

x [`torch.Tensor`, list, or dict] Input variables.

shape [tuple] Shape of samples. If set, *batch_size* and *dim* are ignored.

batch_size [int] Batch size of samples. This is set to 1 by default.

return_all [bool] Choose whether the output contains input variables.

reparam [bool] Choose whether we sample variables with reparameterized trick.

output [dict] Samples of this distribution.

sample_inv (*x, return_all=True, **kwargs*)

log_likelihood (*x*)

Estimate the log likelihood of this distribution from inputs formatted by a dictionary.

x_dict [dict] Input samples.

log_like [`torch.Tensor`] Log-likelihood.

1.5 Operators

1.5.1 ReplaceVarDistribution

class `pixyz.distributions.distributions.ReplaceVarDistribution` (*a, replace_dict*)

Bases: `pixyz.distributions.distributions.Distribution`

Replace names of variables in Distribution.

a [`pixyz.Distribution` (not `pixyz.MultiplyDistribution`)] Distribution.

replace_dict [dict] Dictionary.

forward (**args, **kwargs*)

When this class is inherited by DNNs, it is also intended that this method is overridden.

get_params (*params_dict*)

This method aims to get parameters of this distributions from constant parameters set in initialization and outputs of DNNs.

params_dict [dict] Input parameters.

output_dict [dict] Output parameters

```

>>> print(dist_1.prob_text, dist_1.distribution_name)
p(x) Normal
>>> dist_1.get_params()
{"loc": 0, "scale": 1}
>>> print(dist_2.prob_text, dist_2.distribution_name)
p(x|z) Normal
>>> dist_1.get_params({"z": 1})
{"loc": 0, "scale": 1}

```

sample ($x=\{\}$, $shape=None$, $batch_size=1$, $return_all=True$, $reparam=False$)

Sample variables of this distribution. If *cond_var* is not empty, we should set inputs as a dictionary format.

x [torch.Tensor, list, or dict] Input variables.

shape [tuple] Shape of samples. If set, *batch_size* and *dim* are ignored.

batch_size [int] Batch size of samples. This is set to 1 by default.

return_all [bool] Choose whether the output contains input variables.

reparam [bool] Choose whether we sample variables with reparameterized trick.

output [dict] Samples of this distribution.

log_likelihood (x)

Estimate the log likelihood of this distribution from inputs formatted by a dictionary.

x_dict [dict] Input samples.

log_like [torch.Tensor] Log-likelihood.

sample_mean (x)

input_var

Normally, *input_var* has same values as *cond_var*.

distribution_name

1.5.2 MarginalizeVarDistribution

class pixyz.distributions.distributions.**MarginalizeVarDistribution** (a , *marginalize_list*)

Bases: *pixyz.distributions.distributions.Distribution*

Marginalize variables in Distribution. $p(x) = \int p(x, z) dz$

a [pixyz.Distribution (not pixyz.DistributionBase)] Distribution.

marginalize_list [list] Variables to marginalize.

forward (**args*, ***kwargs*)

When this class is inherited by DNNs, it is also intended that this method is overridden.

get_params (*params_dict*)

This method aims to get parameters of this distributions from constant parameters set in initialization and outputs of DNNs.

params_dict [dict] Input parameters.

output_dict [dict] Output parameters

```
>>> print(dist_1.prob_text, dist_1.distribution_name)
p(x) Normal
>>> dist_1.get_params()
{"loc": 0, "scale": 1}
>>> print(dist_2.prob_text, dist_2.distribution_name)
p(x|z) Normal
>>> dist_1.get_params({"z": 1})
{"loc": 0, "scale": 1}
```

sample ($x=\{\}$, $shape=None$, $batch_size=1$, $return_all=True$, $reparam=False$)

Sample variables of this distribution. If *cond_var* is not empty, we should set inputs as a dictionary format.

x [torch.Tensor, list, or dict] Input variables.

shape [tuple] Shape of samples. If set, *batch_size* and *dim* are ignored.

batch_size [int] Batch size of samples. This is set to 1 by default.

return_all [bool] Choose whether the output contains input variables.

reparam [bool] Choose whether we sample variables with reparameterized trick.

output [dict] Samples of this distribution.

log_likelihood (x)

Estimate the log likelihood of this distribution from inputs formatted by a dictionary.

x_dict [dict] Input samples.

log_like [torch.Tensor] Log-likelihood.

sample_mean (x)

input_var

Normally, *input_var* has same values as *cond_var*.

distribution_name

prob_factorized_text

1.5.3 MultiplyDistribution

class pixyz.distributions.distributions.**MultiplyDistribution** (a, b)

Bases: *pixyz.distributions.distributions.Distribution*

Multiply by given distributions, e.g. $p(x, y|z) = p(x|z, y)p(y|z)$. In this class, it is checked if two distributions can be multiplied.

$p(x|z)p(z|y)$ -> Valid

$p(x|z)p(y|z)$ -> Valid

$p(x|z)p(y|a)$ -> Valid

$p(x|z)p(z|x)$ -> Invalid (recursive)

$p(x|z)p(x|y)$ -> Invalid (conflict)

a [pixyz.Distribution] Distribution.

b [pixyz.Distribution] Distribution.

```
>>> p_multi = MultipleDistribution([a, b])
>>> p_multi = a * b
```

inh_var

input_var

Normally, *input_var* has same values as *cond_var*.

prob_factorized_text

sample (*x*={}, *shape*=None, *batch_size*=1, *return_all*=True, *reparam*=False)

Sample variables of this distribution. If *cond_var* is not empty, we should set inputs as a dictionary format.

x [torch.Tensor, list, or dict] Input variables.

shape [tuple] Shape of samples. If set, *batch_size* and *dim* are ignored.

batch_size [int] Batch size of samples. This is set to 1 by default.

return_all [bool] Choose whether the output contains input variables.

reparam [bool] Choose whether we sample variables with reparameterized trick.

output [dict] Samples of this distribution.

log_likelihood (*x*)

Estimate the log likelihood of this distribution from inputs formatted by a dictionary.

x_dict [dict] Input samples.

log_like [torch.Tensor] Log-likelihood.

1.6 Functions

`pixyz.distributions.distributions.sum_samples` (*samples*)

2.1 Loss

```
class pixyz.losses.losses.Loss (p1, p2=None, input_var=None)
```

Bases: object

input_var

loss_text

mean()

sum()

estimate (x={}, **kwargs)

train (x={}, **kwargs)

Train the implicit (adversarial) loss function.

test (x={}, **kwargs)

Test the implicit (adversarial) loss function.

2.2 Negative expected value of log-likelihood (entropy)

2.2.1 CrossEntropy

```
class pixyz.losses.CrossEntropy (p1, p2, input_var=None)
```

Bases: `pixyz.losses.losses.Loss`

Cross entropy, a.k.a., the negative expected value of log-likelihood (Monte Carlo approximation).

$$-\mathbb{E}_{q(x)}[\log p(x)] \approx -\frac{1}{L} \sum_{l=1}^L \log p(x_l),$$

where $x_l \sim q(x)$.

```
loss_text
estimate (x={})
```

2.2.2 Entropy

```
class pixyz.losses.Entropy (p1, input_var=None)
    Bases: pixyz.losses.losses.Loss
    Entropy (Monte Carlo approximation).
```

$$-\mathbb{E}_{p(x)}[\log p(x)] \approx -\frac{1}{L} \sum_{l=1}^L \log p(x_l),$$

where $x_l \sim p(x)$.

Note: This class is a special case of the *CrossEntropy* class. You can get the same result with *CrossEntropy*.

```
loss_text
estimate (x={})
```

2.2.3 StochasticReconstructionLoss

```
class pixyz.losses.StochasticReconstructionLoss (encoder, decoder, input_var=None)
    Bases: pixyz.losses.losses.Loss
    Reconstruction Loss (Monte Carlo approximation).
```

$$-\mathbb{E}_{q(z|x)}[\log p(x|z)] \approx -\frac{1}{L} \sum_{l=1}^L \log p(x|z_l),$$

where $z_l \sim q(z|x)$.

Note: This class is a special case of the *CrossEntropy* class. You can get the same result with *CrossEntropy*.

```
loss_text
estimate (x={})
```

2.3 Negative log-likelihood

2.3.1 NLL

```
class pixyz.losses.NLL (p, input_var=None)
    Bases: pixyz.losses.losses.Loss
    Negative log-likelihood.
```

$$\log p(x)$$

```
loss_text
estimate (x={})
```

2.4 Lower bound

2.4.1 ELBO

class `pixyz.losses.ELBO` (*p*, *approximate_dist*, *input_var=None*)

Bases: `pixyz.losses.losses.Loss`

The evidence lower bound (Monte Carlo approximation).

$$\mathbb{E}_{q(z|x)}[\log \frac{p(x, z)}{q(z|x)}] \approx \frac{1}{L} \sum_{l=1}^L \log p(x, z_l),$$

where $z_l \sim q(z|x)$.

loss_text

estimate (*x*={}, *batch_size=None*)

2.5 Similarity

2.5.1 SimilarityLoss

class `pixyz.losses.SimilarityLoss` (*p1*, *p2*, *input_var=None*, *var=['z']*, *margin=0*)

Bases: `pixyz.losses.losses.Loss`

Learning Modality-Invariant Representations for Speech and Images (Leidai et. al.)

estimate (*x*)

2.5.2 MultiModalContrastivenessLoss

class `pixyz.losses.MultiModalContrastivenessLoss` (*p1*, *p2*, *input_var=None*, *margin=0.5*)

Bases: `pixyz.losses.losses.Loss`

Disentangling by Partitioning: A Representation Learning Framework for Multimodal Sensory Data

estimate (*x*)

2.6 Adversarial loss (GAN loss)

2.6.1 AdversarialJSDivergence

class `pixyz.losses.AdversarialJSDivergence` (*p_data*, *p*, *discriminator*, *input_var=None*, *optimizer=<class 'torch.optim.adam.Adam'>*, *optimizer_params={}*, *inverse_g_loss=True*)

Bases: `pixyz.losses.losses.Loss`

Adversarial loss (Jensen-Shannon divergence).

$$\mathcal{L}_{adv} = 2JS[p_{data}(x)||p(x)] + const.$$

```

loss_text
estimate (x={}, discriminator=False)
d_loss (y1, y2, batch_size)
g_loss (y1, y2, batch_size)
train (train_x, **kwargs)
    Train the implicit (adversarial) loss function.
test (test_x, **kwargs)
    Test the implicit (adversarial) loss function.

```

2.6.2 AdversarialWassersteinDistance

```

class pixyz.losses.AdversarialWassersteinDistance (p_data, p, discriminator,
                                                    clip_value=0.01, **kwargs)
    Bases: pixyz.losses.adversarial_loss.AdversarialJSDivergence
    Adversarial loss (Wasserstein Distance).

loss_text
d_loss (y1, y2, *args, **kwargs)
g_loss (y1, y2, *args, **kwargs)
train (train_x, **kwargs)
    Train the implicit (adversarial) loss function.

```

2.7 Loss for special purpose

2.7.1 Parameter

```

class pixyz.losses.losses.Parameter (input_var)
    Bases: pixyz.losses.losses.Loss
estimate (x={}, **kwargs)
loss_text

```

2.8 Operators

2.8.1 LossOperator

```

class pixyz.losses.losses.LossOperator (loss1, loss2)
    Bases: pixyz.losses.losses.Loss
loss_text
estimate (x={}, **kwargs)
train (x, **kwargs)
    TODO: Fix

```

```
test (x, **kwargs)
    TODO: Fix
```

2.8.2 LossSelfOperator

```
class pixyz.losses.losses.LossSelfOperator (loss1)
    Bases: pixyz.losses.losses.Loss

    train (x={}, **kwargs)
        Train the implicit (adversarial) loss function.

    test (x={}, **kwargs)
        Test the implicit (adversarial) loss function.
```

2.8.3 AddLoss

```
class pixyz.losses.losses.AddLoss (loss1, loss2)
    Bases: pixyz.losses.losses.LossOperator

    loss_text

    estimate (x={}, **kwargs)
```

2.8.4 SubLoss

```
class pixyz.losses.losses.SubLoss (loss1, loss2)
    Bases: pixyz.losses.losses.LossOperator

    loss_text

    estimate (x={}, **kwargs)
```

2.8.5 MulLoss

```
class pixyz.losses.losses.MulLoss (loss1, loss2)
    Bases: pixyz.losses.losses.LossOperator

    loss_text

    estimate (x={}, **kwargs)
```

2.8.6 DivLoss

```
class pixyz.losses.losses.DivLoss (loss1, loss2)
    Bases: pixyz.losses.losses.LossOperator

    loss_text

    estimate (x={}, **kwargs)
```

2.8.7 NegLoss

```
class pixyz.losses.losses.NegLoss (loss)  
    Bases: pixyz.losses.losses.LossSelfOperator  
  
    loss_text  
  
    estimate (x={}, **kwargs)
```

2.8.8 BatchMean

```
class pixyz.losses.losses.BatchMean (loss)  
    Bases: pixyz.losses.losses.LossSelfOperator  
  
    Loss averaged over batch data.
```

$$\mathbb{E}_{p_{data}(x)}[\mathcal{L}(x)] \approx \frac{1}{N} \sum_{i=1}^N \mathcal{L}(x_i),$$

where $x_i \sim p_{data}(x)$ and \mathcal{L} is a loss function.

```
loss_text  
  
estimate (x={}, **kwargs)
```

2.8.9 BatchSum

```
class pixyz.losses.losses.BatchSum (loss)  
    Bases: pixyz.losses.losses.LossSelfOperator  
  
    Loss summed over batch data.
```

$$\sum_{i=1}^N \mathcal{L}(x_i),$$

where $x_i \sim p_{data}(x)$ and \mathcal{L} is a loss function.

```
loss_text  
  
estimate (x={}, **kwargs)
```

3.1 Model

```
class pixyz.models.Model (loss, test_loss=None, distributions=[], optimizer=<class  
                                'torch.optim.adam.Adam'>, optimizer_params={})  
    Bases: object  
    set_loss (loss, test_loss=None)  
    train (train_x={}, **kwargs)  
    test (test_x={}, **kwargs)
```

3.2 Pre-implementation models

3.2.1 ML

```
class pixyz.models.ML (p, other_distributions=[], optimizer=<class 'torch.optim.adam.Adam'>, opti-  
                                mizer_params={})  
    Bases: pixyz.models.model.Model  
    Maximum Likelihood (log-likelihood)  
    train (train_x={}, **kwargs)  
    test (test_x={}, **kwargs)
```

3.2.2 VAE

```
class pixyz.models.VAE (encoder, decoder, other_distributions=[], regularizer=[], optimizer=<class  
                                'torch.optim.adam.Adam'>, optimizer_params={})  
    Bases: pixyz.models.model.Model  
    Variational Autoencoder
```

[Kingma+ 2013] Auto-Encoding Variational Bayes

train (*train_x*={}, ****kwargs**)

test (*test_x*={}, ****kwargs**)

3.2.3 VI

class pixyz.models.VI (*p*, *approximate_dist*, *other_distributions*=[], *optimizer*=<class 'torch.optim.adam.Adam'>, *optimizer_params*={})

Bases: pixyz.models.model.Model

Variational Inference (Amortized inference)

train (*train_x*={}, ****kwargs**)

test (*test_x*={}, ****kwargs**)

3.2.4 GAN

class pixyz.models.GAN (*p_data*, *p*, *discriminator*, *optimizer*=<class 'torch.optim.adam.Adam'>, *optimizer_params*={}, *d_optimizer*=<class 'torch.optim.adam.Adam'>, *d_optimizer_params*={})

Bases: pixyz.models.model.Model

Generative Adversarial Network

train (*train_x*={}, *adversarial_loss*=True, ****kwargs**)

test (*test_x*={}, *adversarial_loss*=True, ****kwargs**)

CHAPTER 4

pixyz.utils

```
pixyz.utils.set_epsilon(eps)  
pixyz.utils.epsilon()  
pixyz.utils.get_dict_values(dicts, keys, return_dict=False)  
pixyz.utils.delete_dict_values(dicts, keys)  
pixyz.utils.detach_dict(dicts)  
pixyz.utils.replace_dict_keys(dicts, replace_list_dict)  
pixyz.utils.tolist(a)
```


CHAPTER 5

Indices and tables

- `genindex`
- `modindex`
- `search`

p

`pixyz.distributions`, 3

`pixyz.losses`, 13

`pixyz.models`, 19

`pixyz.utils`, 21

A

AddLoss (class in pixyz.losses.losses), 17
 AdversarialJSDivergence (class in pixyz.losses), 15
 AdversarialWassersteinDistance (class in pixyz.losses), 16

B

BatchMean (class in pixyz.losses.losses), 18
 BatchSum (class in pixyz.losses.losses), 18
 Bernoulli (class in pixyz.distributions), 5

C

Categorical (class in pixyz.distributions), 5
 cond_var (pixyz.distributions.distributions.Distribution attribute), 3
 CrossEntropy (class in pixyz.losses), 13
 CustomLikelihoodDistribution (class in pixyz.distributions), 7

D

d_loss() (pixyz.losses.AdversarialJSDivergence method), 16
 d_loss() (pixyz.losses.AdversarialWassersteinDistance method), 16
 DataDistribution (class in pixyz.distributions), 7
 delete_dict_values() (in module pixyz.utils), 21
 detach_dict() (in module pixyz.utils), 21
 Deterministic (class in pixyz.distributions), 6
 Distribution (class in pixyz.distributions.distributions), 3
 distribution_name (pixyz.distributions.Bernoulli attribute), 5
 distribution_name (pixyz.distributions.Categorical attribute), 5
 distribution_name (pixyz.distributions.CustomLikelihoodDistribution attribute), 7
 distribution_name (pixyz.distributions.DataDistribution attribute), 7
 distribution_name (pixyz.distributions.Deterministic attribute), 6

distribution_name (pixyz.distributions.distributions.Distribution attribute), 3
 distribution_name (pixyz.distributions.distributions.MarginalizeVarDistribution attribute), 10
 distribution_name (pixyz.distributions.distributions.ReplaceVarDistribution attribute), 9
 distribution_name (pixyz.distributions.FactorizedBernoulli attribute), 5
 distribution_name (pixyz.distributions.Normal attribute), 4
 distribution_name (pixyz.distributions.RelaxedBernoulli attribute), 5
 distribution_name (pixyz.distributions.RelaxedCategorical attribute), 5
 DivLoss (class in pixyz.losses.losses), 17

E

ELBO (class in pixyz.losses), 15
 Entropy (class in pixyz.losses), 14
 epsilon() (in module pixyz.utils), 21
 estimate() (pixyz.losses.AdversarialJSDivergence method), 16
 estimate() (pixyz.losses.CrossEntropy method), 14
 estimate() (pixyz.losses.ELBO method), 15
 estimate() (pixyz.losses.Entropy method), 14
 estimate() (pixyz.losses.losses.AddLoss method), 17
 estimate() (pixyz.losses.losses.BatchMean method), 18
 estimate() (pixyz.losses.losses.BatchSum method), 18
 estimate() (pixyz.losses.losses.DivLoss method), 17
 estimate() (pixyz.losses.losses.Loss method), 13
 estimate() (pixyz.losses.losses.LossOperator method), 16
 estimate() (pixyz.losses.losses.MulLoss method), 17
 estimate() (pixyz.losses.losses.NegLoss method), 18
 estimate() (pixyz.losses.losses.Parameter method), 16
 estimate() (pixyz.losses.losses.SubLoss method), 17
 estimate() (pixyz.losses.MultiModalContrastivenessLoss method), 15
 estimate() (pixyz.losses.NLL method), 14
 estimate() (pixyz.losses.SimilarityLoss method), 15

estimate() (pixyz.losses.StochasticReconstructionLoss method), 14

experts() (pixyz.distributions.NormalPoE method), 6

F

FactorizedBernoulli (class in pixyz.distributions), 5

forward() (pixyz.distributions.distributions.Distribution method), 4

forward() (pixyz.distributions.distributions.MarginalizeVarDistribution method), 9

forward() (pixyz.distributions.distributions.ReplaceVarDistribution method), 8

forward() (pixyz.distributions.RealNVP method), 8

G

g_loss() (pixyz.losses.AdversarialJSDivergence method), 16

g_loss() (pixyz.losses.AdversarialWassersteinDistance method), 16

GAN (class in pixyz.models), 20

get_dict_values() (in module pixyz.utils), 21

get_params() (pixyz.distributions.distributions.Distribution method), 3

get_params() (pixyz.distributions.distributions.MarginalizeVarDistribution method), 9

get_params() (pixyz.distributions.distributions.ReplaceVarDistribution method), 8

get_params() (pixyz.distributions.NormalPoE method), 6

I

inh_var (pixyz.distributions.distributions.MultiplyDistribution attribute), 11

input_var (pixyz.distributions.CustomLikelihoodDistribution attribute), 7

input_var (pixyz.distributions.DataDistribution attribute), 7

input_var (pixyz.distributions.distributions.Distribution attribute), 3

input_var (pixyz.distributions.distributions.MarginalizeVarDistribution attribute), 10

input_var (pixyz.distributions.distributions.MultiplyDistribution attribute), 11

input_var (pixyz.distributions.distributions.ReplaceVarDistribution attribute), 9

input_var (pixyz.losses.losses.Loss attribute), 13

L

log_likelihood() (pixyz.distributions.CustomLikelihoodDistribution method), 7

log_likelihood() (pixyz.distributions.distributions.Distribution method), 4

log_likelihood() (pixyz.distributions.distributions.MarginalizeVarDistribution method), 10

log_likelihood() (pixyz.distributions.distributions.MultiplyDistribution method), 11

log_likelihood() (pixyz.distributions.distributions.ReplaceVarDistribution method), 9

log_likelihood() (pixyz.distributions.NormalPoE method), 6

log_likelihood() (pixyz.distributions.RealNVP method), 8

log_likelihood() (pixyz.distributions.RelaxedBernoulli method), 5

log_likelihood() (pixyz.distributions.RelaxedCategorical method), 5

Loss (class in pixyz.losses.losses), 13

loss_text (pixyz.losses.AdversarialJSDivergence attribute), 15

loss_text (pixyz.losses.AdversarialWassersteinDistance attribute), 16

loss_text (pixyz.losses.CrossEntropy attribute), 13

loss_text (pixyz.losses.ELBO attribute), 15

loss_text (pixyz.losses.Entropy attribute), 14

loss_text (pixyz.losses.losses.AddLoss attribute), 17

loss_text (pixyz.losses.losses.BatchMean attribute), 18

loss_text (pixyz.losses.losses.BatchSum attribute), 18

loss_text (pixyz.losses.losses.DivLoss attribute), 17

loss_text (pixyz.losses.losses.Loss attribute), 13

loss_text (pixyz.losses.losses.LossOperator attribute), 16

loss_text (pixyz.losses.losses.MulLoss attribute), 17

loss_text (pixyz.losses.losses.NegLoss attribute), 18

loss_text (pixyz.losses.losses.Parameter attribute), 16

loss_text (pixyz.losses.losses.SubLoss attribute), 17

loss_text (pixyz.losses.NLL attribute), 14

loss_text (pixyz.losses.StochasticReconstructionLoss attribute), 14

LossOperator (class in pixyz.losses.losses), 16

LossSelfOperator (class in pixyz.losses.losses), 17

M

marginalize_var() (pixyz.distributions.distributions.Distribution method), 4

MarginalizeVarDistribution (class in pixyz.distributions.distributions), 9

mean() (pixyz.losses.losses.Loss method), 13

ML (class in pixyz.models), 19

Model (class in pixyz.models), 19

MulLoss (class in pixyz.losses.losses), 17

MultiModalContrastivenessLoss (class in pixyz.losses), 15

MultiplyDistribution (class in pixyz.distributions.distributions), 10

N

name (pixyz.distributions.distributions.Distribution attribute), 3

NegLoss (class in pixyz.losses.losses), 18

NLL (class in pixyz.losses), 14
 Normal (class in pixyz.distributions), 4
 NormalPoE (class in pixyz.distributions), 6

P

Parameter (class in pixyz.losses.losses), 16
 pixyz.distributions (module), 3
 pixyz.losses (module), 13
 pixyz.models (module), 19
 pixyz.utils (module), 21
 PlanarFlow (class in pixyz.distributions), 7
 prob_factorized_text (pixyz.distributions.distributions.Distribution attribute), 3
 prob_factorized_text (pixyz.distributions.distributions.MarginalizeVarDistribution attribute), 10
 prob_factorized_text (pixyz.distributions.distributions.MultiplyDistribution attribute), 11
 prob_text (pixyz.distributions.distributions.Distribution attribute), 3
 prob_text (pixyz.distributions.RealNVP attribute), 8

R

RealNVP (class in pixyz.distributions), 8
 RelaxedBernoulli (class in pixyz.distributions), 5
 RelaxedCategorical (class in pixyz.distributions), 5
 replace_dict_keys() (in module pixyz.utils), 21
 replace_var() (pixyz.distributions.distributions.Distribution method), 4
 ReplaceVarDistribution (class in pixyz.distributions.distributions), 8

S

sample() (pixyz.distributions.DataDistribution method), 7
 sample() (pixyz.distributions.Deterministic method), 6
 sample() (pixyz.distributions.distributions.Distribution method), 4
 sample() (pixyz.distributions.distributions.MarginalizeVarDistribution method), 10
 sample() (pixyz.distributions.distributions.MultiplyDistribution method), 11
 sample() (pixyz.distributions.distributions.ReplaceVarDistribution method), 9
 sample() (pixyz.distributions.NormalPoE method), 6
 sample() (pixyz.distributions.RealNVP method), 8
 sample_inv() (pixyz.distributions.RealNVP method), 8
 sample_mean() (pixyz.distributions.Bernoulli method), 5
 sample_mean() (pixyz.distributions.Categorical method), 5
 sample_mean() (pixyz.distributions.distributions.Distribution method), 4
 sample_mean() (pixyz.distributions.distributions.MarginalizeVarDistribution method), 10
 sample_mean() (pixyz.distributions.distributions.ReplaceVarDistribution method), 9

sample_mean() (pixyz.distributions.Normal method), 4
 sample_mean() (pixyz.distributions.NormalPoE method), 6
 sample_mean() (pixyz.distributions.RelaxedBernoulli method), 5
 sample_mean() (pixyz.distributions.RelaxedCategorical method), 6
 set_epsilon() (in module pixyz.utils), 21
 set_loss() (pixyz.models.Model method), 19
 SimilarityLoss (class in pixyz.losses), 15
 StochasticReconstructionLoss (class in pixyz.losses), 14
 SubLoss (class in pixyz.losses.losses), 17
 sum() (pixyz.losses.losses.Loss method), 13
 sum() (pixyz.distributions.distributions.Distribution method), 11

T

test() (pixyz.losses.AdversarialJSDivergence method), 16
 test() (pixyz.losses.losses.Loss method), 13
 test() (pixyz.losses.losses.LossOperator method), 16
 test() (pixyz.losses.losses.LossSelfOperator method), 17
 test() (pixyz.models.GAN method), 20
 test() (pixyz.models.ML method), 19
 test() (pixyz.models.Model method), 19
 test() (pixyz.models.VAE method), 20
 test() (pixyz.models.VI method), 20
 tolist() (in module pixyz.utils), 21
 train() (pixyz.losses.AdversarialJSDivergence method), 16
 train() (pixyz.losses.AdversarialWassersteinDistance method), 16
 train() (pixyz.losses.losses.Loss method), 13
 train() (pixyz.losses.losses.LossOperator method), 16
 train() (pixyz.losses.losses.LossSelfOperator method), 17
 train() (pixyz.models.GAN method), 20
 train() (pixyz.models.ML method), 19
 train() (pixyz.models.Model method), 19
 train() (pixyz.models.VAE method), 20
 train() (pixyz.models.VI method), 20

V

VAE (class in pixyz.models), 19
 var (pixyz.distributions.distributions.Distribution attribute), 3
 VI (class in pixyz.models), 20