dmae

Release 1.0.1

Juan S. Lara

Mar 12, 2022

CONTENTS:

1 Installation **1**
- 1.1 PyPI . . . 1
- 1.2 Source . . . 1
- 1.3 Docker . . . 2

2 dmae **3**
- 2.1 dmae package . . . 3
  - 2.1.1 Submodules . . . 3
  - 2.1.2 dmae.dissimilarities module . . . 3
  - 2.1.3 dmae.initializers module . . . 5
  - 2.1.4 dmae.layers module . . . 7
  - 2.1.5 dmae.losses module . . . 17
  - 2.1.6 dmae.metrics module . . . 19
  - 2.1.7 Module contents . . . 20

3 Indices and tables **21**

Python Module Index **23**

Index **25**

CHAPTER

ONE

INSTALLATION

You can install `dmae` from PyPI using pip, from the source Github repository or pulling a preconfigured docker image.

1.1 PyPI

To install `dmae` using pip you can run the followiing command:

```
pip install dmae
```

(optional) If you have an environment with the nvidia drivers and CUDA, you can instead run:

```
pip install dmae-gpu
```

1.2 Source

You can clone the `dmae` repository as follows:

```
git clone https://github.com/juselara1/dmae.git
```

You must install the requirements:

```
pip install -r requirements.txt
```

(optional) If you have an environment with the nvidia drivers and CUDA, you can instead run:

```
pip install -r requirements-gpu.txt
```

Finally, you can install `dmae` via setuptools

```
pip install --no-deps .
```

1.3 Docker

You can pull a preconfigured docker image with `dmae` from DockerHub:

```
docker pull juselara/dmae:1.1.0
```

(optional) If you have the nvidia drivers installed, you can pull the following image:

```
docker pull juselara/dmae:1.1.0-gpu
```

CHAPTER

TWO

DMAE

2.1 dmae package

2.1.1 Submodules

2.1.2 dmae.dissimilarities module

The *dmae.dissimilarities* module implements several dissimilarity functions in tensorflow.

dmae.dissimilarities.**chebyshev**(*X*, *Y*)
Computes a pairwise Chevyshev distance between two matrices $\max(|\mathbf{x}_i - \mathbf{y}_j|)$.

Parameters

X [array-like, shape=(batch_size, n_features)] Input batch matrix.

Y [array-like, shape=(n_clusters, n_features)] Matrix in which each row represents a centroid of a cluster.

Returns

Z [array-like, shape=(batch_size, n_clusters)] Pairwise dissimilarity matrix.

dmae.dissimilarities.**cosine**(*X*, *Y*)
Computes a pairwise cosine distance between two matrices $\mathbf{D}_{ij} = (\mathbf{x}_i \cdot \mathbf{y}_j)/(||\mathbf{x}_i|| \cdot ||\mathbf{y}_j||)$.

Parameters

X [array-like, shape=(batch_size, n_features)] Input batch matrix.

Y [array-like, shape=(n_clusters, n_features)] Matrix in which each row represents a centroid of a cluster.

Returns

Z [array-like, shape=(batch_size, n_clusters)] Pairwise dissimilarity matrix.

dmae.dissimilarities.**euclidean**(*X*, *Y*)
Computes a pairwise Euclidean distance between two matrices $\mathbf{D}_{ij} = ||\mathbf{x}_i - \mathbf{y}_j||$.

Parameters

X [array-like, shape=(batch_size, n_features)] Input batch matrix.

Y [array-like, shape=(n_clusters, n_features)] Matrix in which each row represents a centroid of a cluster.

Returns

Z [array-like, shape=(batch_size, n_clusters)] Pairwise dissimilarity matrix.

dmae.dissimilarities.**kullback_leibler**(*loggit_P*, *loggit_Q*, *eps=0.001*, *normalization='softmax_abs'*)

Kullback Leibler divergence. $\sum_x P_x \log P_x - P_x \log Q_x$

Parameters

loggit_P [array-like, shape=(batch_size, n_features)] Input batch matrix of loggits.

loggit_Q [array-like, shape=(n_features, n_features)] Matrix in which each row represents the unsigned loggit of a cluster.

eps: float, default=1e-3 Hyperparameter to avoid numerical issues.

normalization: {str, function}, default="softmax_abs" Specifies which normalization function is used to transform the data into probabilities. You can specify a custom functon *f(X, eps)* with the arguments *X* and *eps*, or use a predefined function {"softmax_abs", "softmax_relu", "squared_sum", "abs_sum", "relu_sum", "identity"}

Returns

Z [array-like, shape=(batch_size, n_clusters)] Pairwise dissimilarity matrix.

dmae.dissimilarities.**mahalanobis**(*X*, *Y*, *cov*)

Computes a pairwise Mahalanobis distance $\mathbf{D}_{ij} = (\mathbf{x}_i - \mathbf{y}_j)^T \Sigma_j (\mathbf{x}_i - \mathbf{y}_j)$.

Parameters

X [array-like, shape=(batch_size, n_features)] Input batch matrix.

Y [array-like, shape=(n_features, n_features)] Matrix in which each row represents a centroid of a cluster.

cov: array-like, shape=(n_clusters, n_features, n_features) 3D Tensor with the inverse covariance matrices of all the clusters.

Returns

Z [array-like, shape=(batch_size, n_clusters)] Pairwise dissimilarity matrix.

dmae.dissimilarities.**manhattan**(*X*, *Y*)

Computes a pairwise Manhattan distance between two matrices $\mathbf{D}_{ij} = \sum |\mathbf{x}_i| - |\mathbf{y}_j|$.

Parameters

X [array-like, shape=(batch_size, n_features)] Input batch matrix.

Y [array-like, shape=(n_clusters, n_features)] Matrix in which each row represents a centroid of a cluster.

Returns

Z [array-like, shape=(batch_size, n_clusters)] Pairwise dissimilarity matrix.

dmae.dissimilarities.**minkowsky**(*X*, *Y*, *p*)

Computes a pairwise Minkowsky distance between two matrices $\mathbf{D}_{ij} = (\sum |\mathbf{x}_i - \mathbf{y}_j|^p)^{1/p}$.

Parameters

X [array-like, shape=(batch_size, n_features)] Input batch matrix.

Y [array-like, shape=(n_clusters, n_features)] Matrix in which each row represents a centroid of a cluster.

p [float] Order of the Minkowsky distance.

Returns

Z [array-like, shape=(batch_size, n_clusters)] Pairwise dissimilarity matrix.

dmae.dissimilarities.**toroidal_euclidean**(*X, Y, interval=<tf.Tensor: shape=(2,), dtype=float32, numpy=array([2., 2.], dtype=float32)>*)

Euclidean dissimilarity that considers circular boundaries.

Parameters

X [array-like, shape=(batch_size, n_features)] Input batch matrix.

Y [array-like, shape=(n_features, n_features)] Matrix in which each row represents a centroid of a cluster.

interval [array-like, default=tf.constant((2.0, 2.0))] Array representing the range on each axis.

Returns

Z [array-like, shape=(batch_size, n_clusters)] Pairwise dissimilarity matrix.

2.1.3 dmae.initializers module

The *dmae.initializers* module implements some initializers for DMAE.

class dmae.initializers.**InitIdentityCov**(*X*, *n_clusters*)

Bases: tensorflow.python.keras.initializers.initializers_v2.Initializer

A tf.keras initializer to assign identity matrices to the covariance parameters.

Parameters

X: array-like, shape=(n_samples, n_features) Input data.

n_clusters: int Number of clusters.

Methods

__call__(shape, dtype)	Generates identity matrices for the given shape and type.
from_config(config)	Instantiates an initializer from a configuration dictionary.
get_config()	Returns the configuration of the initializer as a JSON-serializable dict.

class dmae.initializers.**InitKMeans**(*kmeans_model*)

Bases: tensorflow.python.keras.initializers.initializers_v2.Initializer

A tf.keras initializer to assign the clusters from a sklearn's KMeans model.

Parameters

kmeans_model: :mod:`sklearn.cluster.KMeans` Pretrained KMeans model to initialize DMAE.

Methods

`__call__`(shape, dtype)	Converts KMeans centroids into tensors.
`from_config`(config)	Instantiates an initializer from a configuration dictionary.
`get_config`()	Returns the configuration of the initializer as a JSON-serializable dict.

class dmae.initializers.**InitKMeansCov**(*kmeans_model*, *X*, *n_clusters*)

Bases: tensorflow.python.keras.initializers.initializers_v2.Initializer

A tf.keras initializer to compute covariance matrices from K-means.

Parameters

kmeans_model: :mod:`sklearn.cluster.KMeans` Pretrained KMeans model to initialize DMAE.

X: array-like, shape=(n_samples, n_features) Input data.

n_clusters: int Number of clusters.

Methods

`__call__`(shape, dtype)	Computes covariance matrices from the KMeans predictions.
`from_config`(config)	Instantiates an initializer from a configuration dictionary.
`get_config`()	Returns the configuration of the initializer as a JSON-serializable dict.

class dmae.initializers.**InitPlusPlus**(*X*, *n_clusters*, *dissimilarity=<function euclidean>*, *iters=100*)

Bases: tensorflow.python.keras.initializers.initializers_v2.Initializer

A tf.keras initializer based on K-Means++ that allows dissimilarities.

Parameters

X: array-like, shape=(n_samples, n_features) Input data.

n_clusters: int Number of clusters.

dissimilarity: function, default: :mod:`dmae.dissimilarities.euclidean` A tensorflow function that computes a paiwise dissimilarity function between a batch of points and the cluster's parameters.

iters: int, default: 100 Number of interations to run the K-means++ initialization.

Methods

`__call__`(shape, dtype)	Estimates *n_clusters* using K-means++
`from_config`(config)	Instantiates an initializer from a configuration dictionary.
`get_config`()	Returns the configuration of the initializer as a JSON-serializable dict.

2.1.4 dmae.layers module

The *dmae.layers* module implements the dissimilarity mixture autoencoder (DMAE) layers as tensorflow keras layers.

class dmae.layers.**DissimilarityMixtureAutoencoder**(**args*, ***kwargs*)

Bases: tensorflow.python.keras.engine.base_layer.Layer

A `tf.keras` layer with the Dissimilarity Mixture Autoencoder (DMAE).

Parameters

alpha [float] Softmax inverse temperature.

n_clusters [int] Number of clusters.

dissimilarity [function, default = *dmae.dissimilarities.euclidean*] A tensorflow function that computes a pairwise dissimilarity function between a batch of points and the cluster's parameters.

trainable [dict, default = {"centers": True, "mixers": True}] Specifies which parameters are trainable.

initializers [dict, default = {"centers": `RandomUniform(-1, 1)`, "mixers": `Constant(1.0)`}] Specifies a keras initializer (`tf.keras.initializers`) for each parameter.

regularizers [dict, default = {"centers": None, "mixers": None}] Specifies a keras regularizer (`tf.keras.regularizers`) for each parameter.

Attributes

activity_regularizer Optional regularizer function for the output of this layer.

compute_dtype The dtype of the layer's computations.

dtype The dtype of the layer weights.

dtype_policy The dtype policy associated with this layer.

dynamic Whether the layer is dynamic (eager-only); set in the constructor.

inbound_nodes Deprecated, do NOT use! Only for compatibility with external Keras.

input Retrieves the input tensor(s) of a layer.

input_mask Retrieves the input mask tensor(s) of a layer.

input_shape Retrieves the input shape(s) of a layer.

input_spec *InputSpec* instance(s) describing the input format for this layer.

losses List of losses added using the *add_loss()* API.

metrics List of metrics added using the *add_metric()* API.

name Name of the layer (string), set in the constructor.

name_scope Returns a *tf.name_scope* instance for this class.

non_trainable_variables

non_trainable_weights List of all non-trainable weights tracked by this layer.

outbound_nodes Deprecated, do NOT use! Only for compatibility with external Keras.

output Retrieves the output tensor(s) of a layer.

output_mask Retrieves the output mask tensor(s) of a layer.

output_shape Retrieves the output shape(s) of a layer.

stateful

submodules Sequence of all sub-modules.

supports_masking Whether this layer supports computing a mask using *compute_mask*.

trainable

trainable_variables Sequence of trainable variables owned by this module and its sub-modules.

trainable_weights List of all trainable weights tracked by this layer.

updates

variable_dtype Alias of *Layer.dtype*, the dtype of the weights.

variables Returns the list of all layer variables/weights.

weights Returns the list of all layer variables/weights.

Methods

`__call__`(*args, **kwargs)	Wraps *call*, applying pre- and post-processing steps.
`add_loss`(losses, **kwargs)	Add loss tensor(s), potentially dependent on layer inputs.
`add_metric`(value[, name])	Adds metric tensor to the layer.
`add_update`(updates[, inputs])	Add update op(s), potentially dependent on layer inputs.
`add_variable`(*args, **kwargs)	Deprecated, do NOT use! Alias for *add_weight*.
`add_weight`([name, shape, dtype, ...])	Adds a new variable to the layer.
`apply`(inputs, *args, **kwargs)	Deprecated, do NOT use!
build(input_shape)	Builds the tensorflow variables.
call(x)	Forward pass in DMAE.
`compute_mask`(inputs[, mask])	Computes an output mask tensor.
`compute_output_shape`(input_shape)	Computes the output shape of the layer.
`compute_output_signature`(input_signature)	Compute the output tensor signature of the layer based on the inputs.
`count_params`()	Count the total number of scalars composing the weights.
`from_config`(config)	Creates a layer from its config.
`get_config`()	Returns the config of the layer.

continues on next page

Table 5 – continued from previous page

get_input_at(node_index)	Retrieves the input tensor(s) of a layer at a given node.
get_input_mask_at(node_index)	Retrieves the input mask tensor(s) of a layer at a given node.
get_input_shape_at(node_index)	Retrieves the input shape(s) of a layer at a given node.
get_losses_for(inputs)	Deprecated, do NOT use!
get_output_at(node_index)	Retrieves the output tensor(s) of a layer at a given node.
get_output_mask_at(node_index)	Retrieves the output mask tensor(s) of a layer at a given node.
get_output_shape_at(node_index)	Retrieves the output shape(s) of a layer at a given node.
get_updates_for(inputs)	Deprecated, do NOT use!
get_weights()	Returns the current weights of the layer.
set_weights(weights)	Sets the weights of the layer, from Numpy arrays.
with_name_scope(method)	Decorator to automatically enter the module name scope.

build(*input_shape*)
Builds the tensorflow variables.

Parameters

input_shape [tuple] Input tensor shape.

call(*x*)
Forward pass in DMAE.

Parameters

x [array_like] Input tensor.

Returns

mu_tilde [array_like] Soft-assigned centroids.

pi_tilde [array_like] Soft-assigned mixing coefficients.

class dmae.layers.**DissimilarityMixtureAutoencoderCov**(**args*, ***kwargs*)
Bases: tensorflow.python.keras.engine.base_layer.Layer

A tf.keras layer with the Dissimilarity Mixture Autoencoder (DMAE). This layer includes a covariance parameter for dissimilarities that allow it.

Parameters

alpha [float] Softmax inverse temperature.

n_clusters [int] Number of clusters.

dissimilarity [function, default = *dmae.dissimilarities.mahalanobis*] A tensorflow function that computes a pairwise dissimilarity function between a batch of points and the cluster's parameters.

trainable [dict, default = {"centers": True, "cov": True, mixers": True}] Specifies which parameters are trainable.

initializers [dict, default = {"centers": RandomUniform(-1, 1), "cov": RandomUniform(-1, 1)]

"mixers": :mod:`Constant(1.0)`} Specifies a keras initializer (`tf.keras.initializers`) for each parameter.

regularizers [dict, default = {"centers": None, "cov": None, "mixers": None}] Specifies a keras regularizer (`tf.keras.regularizers`) for each parameter.

Attributes

activity_regularizer Optional regularizer function for the output of this layer.

compute_dtype The dtype of the layer's computations.

dtype The dtype of the layer weights.

dtype_policy The dtype policy associated with this layer.

dynamic Whether the layer is dynamic (eager-only); set in the constructor.

inbound_nodes Deprecated, do NOT use! Only for compatibility with external Keras.

input Retrieves the input tensor(s) of a layer.

input_mask Retrieves the input mask tensor(s) of a layer.

input_shape Retrieves the input shape(s) of a layer.

input_spec *InputSpec* instance(s) describing the input format for this layer.

losses List of losses added using the *add_loss()* API.

metrics List of metrics added using the *add_metric()* API.

name Name of the layer (string), set in the constructor.

name_scope Returns a *tf.name_scope* instance for this class.

non_trainable_variables

non_trainable_weights List of all non-trainable weights tracked by this layer.

outbound_nodes Deprecated, do NOT use! Only for compatibility with external Keras.

output Retrieves the output tensor(s) of a layer.

output_mask Retrieves the output mask tensor(s) of a layer.

output_shape Retrieves the output shape(s) of a layer.

stateful

submodules Sequence of all sub-modules.

supports_masking Whether this layer supports computing a mask using *compute_mask*.

trainable

trainable_variables Sequence of trainable variables owned by this module and its sub-modules.

trainable_weights List of all trainable weights tracked by this layer.

updates

variable_dtype Alias of *Layer.dtype*, the dtype of the weights.

variables Returns the list of all layer variables/weights.

weights Returns the list of all layer variables/weights.

Methods

`__call__`(*args, **kwargs)	Wraps *call*, applying pre- and post-processing steps.
`add_loss`(losses, **kwargs)	Add loss tensor(s), potentially dependent on layer inputs.
`add_metric`(value[, name])	Adds metric tensor to the layer.
`add_update`(updates[, inputs])	Add update op(s), potentially dependent on layer inputs.
`add_variable`(*args, **kwargs)	Deprecated, do NOT use! Alias for *add_weight*.
`add_weight`([name, shape, dtype, ...])	Adds a new variable to the layer.
`apply`(inputs, *args, **kwargs)	Deprecated, do NOT use!
`build`(input_shape)	Builds the tensorflow variables.
`call`(x)	Forward pass in DMAE.
`compute_mask`(inputs[, mask])	Computes an output mask tensor.
`compute_output_shape`(input_shape)	Computes the output shape of the layer.
`compute_output_signature`(input_signature)	Compute the output tensor signature of the layer based on the inputs.
`count_params`()	Count the total number of scalars composing the weights.
`from_config`(config)	Creates a layer from its config.
`get_config`()	Returns the config of the layer.
`get_input_at`(node_index)	Retrieves the input tensor(s) of a layer at a given node.
`get_input_mask_at`(node_index)	Retrieves the input mask tensor(s) of a layer at a given node.
`get_input_shape_at`(node_index)	Retrieves the input shape(s) of a layer at a given node.
`get_losses_for`(inputs)	Deprecated, do NOT use!
`get_output_at`(node_index)	Retrieves the output tensor(s) of a layer at a given node.
`get_output_mask_at`(node_index)	Retrieves the output mask tensor(s) of a layer at a given node.
`get_output_shape_at`(node_index)	Retrieves the output shape(s) of a layer at a given node.
`get_updates_for`(inputs)	Deprecated, do NOT use!
`get_weights`()	Returns the current weights of the layer.
`set_weights`(weights)	Sets the weights of the layer, from Numpy arrays.
`with_name_scope`(method)	Decorator to automatically enter the module name scope.

build(*input_shape*)
Builds the tensorflow variables.

Parameters

input_shape [tuple] Input tensor shape.

call(*x*)
Forward pass in DMAE.

Parameters

x [array_like] Input tensor.

Returns

mu_tilde [array_like] Soft-assigned centroids.

Cov_hat [array_like] Soft-assigned covariance matrices.

pi_tilde [array_like] Soft-assigned mixing coefficients.

class dmae.layers.**DissimilarityMixtureEncoder**(**args*, ***kwargs*)

Bases: tensorflow.python.keras.engine.base_layer.Layer

A tf.keras layer that implements the dissimilarity mixture encoder (DM-Encoder). It computes the soft assignments using a dissimilarity function from *dmae.dissimilarities*.

Parameters

alpha [float] Softmax inverse temperature.

n_clusters [int] Number of clusters.

dissimilarity [function, default = *dmae.dissimilarities.euclidean*] A tensorflow function that computes a pairwise dissimilarity function between a batch of points and the cluster's parameters.

trainable [dict, default = {"centers": True, "mixers": True}] Specifies which parameters are trainable.

initializers [dict, default = {"centers": RandomUniform(-1, 1), "mixers": Constant(1.0)}] Specifies a keras initializer (tf.keras.initializers) for each parameter.

regularizers [dict, default = {"centers": None, "mixers": None}] Specifies a keras regularizer (tf.keras.regularizers) for each parameter.

Attributes

activity_regularizer Optional regularizer function for the output of this layer.

compute_dtype The dtype of the layer's computations.

dtype The dtype of the layer weights.

dtype_policy The dtype policy associated with this layer.

dynamic Whether the layer is dynamic (eager-only); set in the constructor.

inbound_nodes Deprecated, do NOT use! Only for compatibility with external Keras.

input Retrieves the input tensor(s) of a layer.

input_mask Retrieves the input mask tensor(s) of a layer.

input_shape Retrieves the input shape(s) of a layer.

input_spec *InputSpec* instance(s) describing the input format for this layer.

losses List of losses added using the *add_loss()* API.

metrics List of metrics added using the *add_metric()* API.

name Name of the layer (string), set in the constructor.

name_scope Returns a *tf.name_scope* instance for this class.

non_trainable_variables

non_trainable_weights List of all non-trainable weights tracked by this layer.

outbound_nodes Deprecated, do NOT use! Only for compatibility with external Keras.

output Retrieves the output tensor(s) of a layer.

output_mask Retrieves the output mask tensor(s) of a layer.

output_shape Retrieves the output shape(s) of a layer.

stateful

submodules Sequence of all sub-modules.

supports_masking Whether this layer supports computing a mask using *compute_mask*.

trainable

trainable_variables Sequence of trainable variables owned by this module and its sub-modules.

trainable_weights List of all trainable weights tracked by this layer.

updates

variable_dtype Alias of *Layer.dtype*, the dtype of the weights.

variables Returns the list of all layer variables/weights.

weights Returns the list of all layer variables/weights.

Methods

`__call__`(*args, **kwargs)	Wraps *call*, applying pre- and post-processing steps.
`add_loss`(losses, **kwargs)	Add loss tensor(s), potentially dependent on layer inputs.
`add_metric`(value[, name])	Adds metric tensor to the layer.
`add_update`(updates[, inputs])	Add update op(s), potentially dependent on layer inputs.
`add_variable`(*args, **kwargs)	Deprecated, do NOT use! Alias for *add_weight*.
`add_weight`([name, shape, dtype, ...])	Adds a new variable to the layer.
`apply`(inputs, *args, **kwargs)	Deprecated, do NOT use!
`build`(input_shape)	Builds the tensorflow variables.
`call`(x)	Forward pass in DM-Encoder.
`compute_mask`(inputs[, mask])	Computes an output mask tensor.
`compute_output_shape`(input_shape)	Computes the output shape of the layer.
`compute_output_signature`(input_signature)	Compute the output tensor signature of the layer based on the inputs.
`count_params`()	Count the total number of scalars composing the weights.
`from_config`(config)	Creates a layer from its config.
`get_config`()	Returns the config of the layer.
`get_input_at`(node_index)	Retrieves the input tensor(s) of a layer at a given node.
`get_input_mask_at`(node_index)	Retrieves the input mask tensor(s) of a layer at a given node.
`get_input_shape_at`(node_index)	Retrieves the input shape(s) of a layer at a given node.
`get_losses_for`(inputs)	Deprecated, do NOT use!
`get_output_at`(node_index)	Retrieves the output tensor(s) of a layer at a given node.

continues on next page

Table 7 – continued from previous page

get_output_mask_at(node_index)	Retrieves the output mask tensor(s) of a layer at a given node.
get_output_shape_at(node_index)	Retrieves the output shape(s) of a layer at a given node.
get_updates_for(inputs)	Deprecated, do NOT use!
get_weights()	Returns the current weights of the layer.
set_weights(weights)	Sets the weights of the layer, from Numpy arrays.
with_name_scope(method)	Decorator to automatically enter the module name scope.

build(*input_shape*)
Builds the tensorflow variables.

Parameters

input_shape [tuple] Input tensor shape.

call(*x*)
Forward pass in DM-Encoder.

Parameters

x [array_like] Input tensor.

Returns

S [array_like] Soft assignments.

class dmae.layers.**DissimilarityMixtureEncoderCov**(**args*, ***kwargs*)
Bases: tensorflow.python.keras.engine.base_layer.Layer

A tf.keras layer that implements the dissimilarity mixture encoder (DM-Encoder). It computes the soft assignments using a dissimilarity function from *dmae.dissimilarities*. This layer includes a covariance parameter for dissimilarities that allow it.

Parameters

alpha [float] Softmax inverse temperature.

n_clusters [int] Number of clusters.

dissimilarity [function, default = *dmae.dissimilarities.mahalanobis*] A tensorflow function that computes a pairwise dissimilarity function between a batch of points and the cluster's parameters.

trainable [dict, default = {"centers": True, "cov": True, mixers": True}] Specifies which parameters are trainable.

initializers [dict, default = {"centers": RandomUniform(-1, 1), "cov": RandomUniform(-1, 1)]

"mixers": :mod:`Constant(1.0)`} Specifies a keras initializer (tf.keras.initializers) for each parameter.

regularizers [dict, default = {"centers": None, "cov": None, "mixers": None}] Specifies a keras regularizer (tf.keras.regularizers) for each parameter.

Attributes

activity_regularizer Optional regularizer function for the output of this layer.

compute_dtype The dtype of the layer's computations.

dtype The dtype of the layer weights.

dtype_policy The dtype policy associated with this layer.

dynamic Whether the layer is dynamic (eager-only); set in the constructor.

inbound_nodes Deprecated, do NOT use! Only for compatibility with external Keras.

input Retrieves the input tensor(s) of a layer.

input_mask Retrieves the input mask tensor(s) of a layer.

input_shape Retrieves the input shape(s) of a layer.

input_spec *InputSpec* instance(s) describing the input format for this layer.

losses List of losses added using the *add_loss()* API.

metrics List of metrics added using the *add_metric()* API.

name Name of the layer (string), set in the constructor.

name_scope Returns a *tf.name_scope* instance for this class.

non_trainable_variables

non_trainable_weights List of all non-trainable weights tracked by this layer.

outbound_nodes Deprecated, do NOT use! Only for compatibility with external Keras.

output Retrieves the output tensor(s) of a layer.

output_mask Retrieves the output mask tensor(s) of a layer.

output_shape Retrieves the output shape(s) of a layer.

stateful

submodules Sequence of all sub-modules.

supports_masking Whether this layer supports computing a mask using *compute_mask*.

trainable

trainable_variables Sequence of trainable variables owned by this module and its sub-modules.

trainable_weights List of all trainable weights tracked by this layer.

updates

variable_dtype Alias of *Layer.dtype*, the dtype of the weights.

variables Returns the list of all layer variables/weights.

weights Returns the list of all layer variables/weights.

Methods

`__call__`(*args, **kwargs)	Wraps *call*, applying pre- and post-processing steps.
`add_loss`(losses, **kwargs)	Add loss tensor(s), potentially dependent on layer inputs.
`add_metric`(value[, name])	Adds metric tensor to the layer.
`add_update`(updates[, inputs])	Add update op(s), potentially dependent on layer inputs.
`add_variable`(*args, **kwargs)	Deprecated, do NOT use! Alias for *add_weight*.
`add_weight`([name, shape, dtype, ...])	Adds a new variable to the layer.
`apply`(inputs, *args, **kwargs)	Deprecated, do NOT use!
`build`(input_shape)	Builds the tensorflow variables.
`call`(x)	Forward pass in DM-Encoder.
`compute_mask`(inputs[, mask])	Computes an output mask tensor.
`compute_output_shape`(input_shape)	Computes the output shape of the layer.
`compute_output_signature`(input_signature)	Compute the output tensor signature of the layer based on the inputs.
`count_params`()	Count the total number of scalars composing the weights.
`from_config`(config)	Creates a layer from its config.
`get_config`()	Returns the config of the layer.
`get_input_at`(node_index)	Retrieves the input tensor(s) of a layer at a given node.
`get_input_mask_at`(node_index)	Retrieves the input mask tensor(s) of a layer at a given node.
`get_input_shape_at`(node_index)	Retrieves the input shape(s) of a layer at a given node.
`get_losses_for`(inputs)	Deprecated, do NOT use!
`get_output_at`(node_index)	Retrieves the output tensor(s) of a layer at a given node.
`get_output_mask_at`(node_index)	Retrieves the output mask tensor(s) of a layer at a given node.
`get_output_shape_at`(node_index)	Retrieves the output shape(s) of a layer at a given node.
`get_updates_for`(inputs)	Deprecated, do NOT use!
`get_weights`()	Returns the current weights of the layer.
`set_weights`(weights)	Sets the weights of the layer, from Numpy arrays.
`with_name_scope`(method)	Decorator to automatically enter the module name scope.

build(*input_shape*)
Builds the tensorflow variables.

Parameters

input_shape [tuple] Input tensor shape.

call(*x*)
Forward pass in DM-Encoder.

Parameters

x [array_like] Input tensor.

Returns

S [array_like] Soft assignments.

2.1.5 dmae.losses module

The *dmae.losses* module implements several loss functions for each dissimilarity in *dmae.dissimilarities*.

dmae.losses.**chebyshev_loss**(*X*, *mu_tilde*, *pi_tilde*, *alpha*)
Computes the Chebyshev loss.

Parameters

X: array-like, shape=(batch_size, n_features) Input batch matrix.

mu_tilde: array-like, shape=(batch_size, n_features) Matrix in which each row represents the assigned mean vector.

pi_tilde: array-like, shape=(batch_size,) Vector in which each element represents the assigned mixing coefficient.

alpha: float Softmax inverse temperature.

Returns

loss: float Computed loss for each sample.

dmae.losses.**cosine_loss**(*X*, *mu_tilde*, *pi_tilde*, *alpha*)
Computes the cosine loss.

Parameters

X: array-like, shape=(batch_size, n_features) Input batch matrix.

mu_tilde: array-like, shape=(batch_size, n_features) Matrix in which each row represents the assigned mean vector.

pi_tilde: array-like, shape=(batch_size,) Vector in which each element represents the assigned mixing coefficient.

alpha: float Softmax inverse temperature.

Returns

loss: array-like, shape=(batch_size,) Computed loss for each sample.

dmae.losses.**euclidean_loss**(*X*, *mu_tilde*, *pi_tilde*, *alpha*)
Computes the Euclidean loss.

Parameters

X: array-like, shape=(batch_size, n_features) Input batch matrix.

mu_tilde: array-like, shape=(batch_size, n_features) Matrix in which each row represents the assigned mean vector.

pi_tilde: array-like, shape=(batch_size,) Vector in which each element represents the assigned mixing coefficient.

alpha: float Softmax inverse temperature.

Returns

loss: array-like, shape=(batch_size,) Computed loss for each sample.

dmae.losses.**kullback_leibler_loss**(*loggit_P*, *loggit_Q_tilde*, *pi_tilde*, *alpha*, *eps=0.001*, *normalization='softmax_abs'*)

Loss for the Kullback Leibler divergence.

Parameters

loggit_P: array-like, shape=(batch_size, n_features) Input batch loggits (pre-normalization values).

loggit_Q_tilde: array-like, shape=(batch_size, n_features) Cluster loggits (pre-normalization values)

pi_tilde: array-like, shape=(batch_size,) Vector in which each element represents the assigned mixing coefficient.

alpha: float Softmax inverse temperature.

normalization: {str, function}, default="softmax_abs" Specifies which normalization function is used to transform the data into probabilities. You can specify a custom functon *f(X, eps)* with the arguments *X* and *eps*, or use a predefined function {"softmax_abs", "softmax_relu", "squared_sum", "abs_sum", "relu_sum", "identity"}

Returns

loss: float Computed loss for each batch.

dmae.losses.**mahalanobis_loss**(*X*, *mu_tilde*, *Cov_tilde*, *pi_tilde*, *alpha*)

Computes the Mahalanobis loss.

Parameters

X: array-like, shape=(batch_size, n_features) Input batch matrix.

mu_tilde: array-like, shape=(batch_size, n_features) Matrix in which each row represents the assigned mean vector.

Cov_tilde: array-like, shape=(batch_size, n_features, n_features) Tensor with the assigned covariances.

pi_tilde: array-like, shape=(batch_size,) Vector in which each element represents the assigned mixing coefficient.

alpha: float Softmax inverse temperature.

Returns

loss: array-like, shape=(batch_size,) Computed loss for each sample.

dmae.losses.**manhattan_loss**(*X*, *mu_tilde*, *pi_tilde*, *alpha*)

Computes the Manhattan loss.

Parameters

X: array-like, shape=(batch_size, n_features) Input batch matrix.

mu_tilde: array-like, shape=(batch_size, n_features) Matrix in which each row represents the assigned mean vector.

pi_tilde: array-like, shape=(batch_size,) Vector in which each element represents the assigned mixing coefficient.

alpha: float Softmax inverse temperature.

Returns

loss: array-like, shape=(batch_size,) Computed loss for each sample.

dmae.losses.**minkowsky_loss**(*X*, *mu_tilde*, *pi_tilde*, *alpha*, *p*)
Computes the Minkowsky loss.

Parameters

X: array-like, shape=(batch_size, n_features) Input batch matrix.

mu_tilde: array-like, shape=(batch_size, n_features) Matrix in which each row represents the assigned mean vector.

pi_tilde: array-like, shape=(batch_size,) Vector in which each element represents the assigned mixing coefficient.

alpha: float Softmax inverse temperature.

p: float Order of the Minkowsky distance

Returns

loss: array-like, shape=(batch_size,) Computed loss for each sample.

dmae.losses.**toroidal_euclidean_loss**(*X, mu_tilde, pi_tilde, alpha, interval=<tf.Tensor: shape=(2,), dtype=float32, numpy=array([2., 2.], dtype=float32)>*)
Loss for the toroidal euclidean dissimilarity.

Parameters

X: array-like, shape=(batch_size, n_features) Input batch matrix.

mu_tilde: array-like, shape=(batch_size, n_features) Matrix in which each row represents the assigned mean vector.

pi_tilde: array-like, shape=(batch_size,) Vector in which each element represents the assigned mixing coefficient.

alpha: float Softmax inverse temperature.

interval [array-like, default=tf.constant((2.0, 2.0))] Array representing the range on each axis.

Returns

loss: float Computed loss for each batch.

2.1.6 dmae.metrics module

The *dmae.metrics* module implements some evaluation metrics that are used in the paper.

dmae.metrics.**unsupervised_classification_accuracy**(*y_true*, *y_pred*)
Scipy-based implementation of the unsupervised classification accuracy.

Parameters

y_true: array-like, shape=(n_samples,) Array with the Ground truth labels.

y_pred: array-like, shape=(n_samples,) Array with the predicted labels.

Returns

uacc: float Unsupervised classification accuracy between y_true and y_pred.

dmae.metrics.**zero_norm**(*preds*, *thr=1e-07*)
Numpy implementation of the L0 norm.

Parameters

preds: array-like, shape=(n_samples, n_clusters) Soft-assignments extracted from a DM-Encoder

thr: float Threshold used to compute the L0 norm.

Returns

L0: float L0 norm of the soft-assignments.

2.1.7 Module contents

CHAPTER

THREE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

d

dmae, 20
dmae.dissimilarities, 3
dmae.initializers, 5
dmae.layers, 7
dmae.losses, 17
dmae.metrics, 19

INDEX

B

build() (*dmae.layers.DissimilarityMixtureAutoencoder method*), 9
build() (*dmae.layers.DissimilarityMixtureAutoencoderCov method*), 11
build() (*dmae.layers.DissimilarityMixtureEncoder method*), 14
build() (*dmae.layers.DissimilarityMixtureEncoderCov method*), 16

C

call() (*dmae.layers.DissimilarityMixtureAutoencoder method*), 9
call() (*dmae.layers.DissimilarityMixtureAutoencoderCov method*), 11
call() (*dmae.layers.DissimilarityMixtureEncoder method*), 14
call() (*dmae.layers.DissimilarityMixtureEncoderCov method*), 16
chebyshev() (*in module dmae.dissimilarities*), 3
chebyshev_loss() (*in module dmae.losses*), 17
cosine() (*in module dmae.dissimilarities*), 3
cosine_loss() (*in module dmae.losses*), 17

D

DissimilarityMixtureAutoencoder (*class in dmae.layers*), 7
DissimilarityMixtureAutoencoderCov (*class in dmae.layers*), 9
DissimilarityMixtureEncoder (*class in dmae.layers*), 12
DissimilarityMixtureEncoderCov (*class in dmae.layers*), 14
dmae
    module, 20
dmae.dissimilarities
    module, 3
dmae.initializers
    module, 5
dmae.layers
    module, 7
dmae.losses
    module, 17
dmae.metrics
    module, 19

E

euclidean() (*in module dmae.dissimilarities*), 3
euclidean_loss() (*in module dmae.losses*), 17

I

InitIdentityCov (*class in dmae.initializers*), 5
InitKMeans (*class in dmae.initializers*), 5
InitKMeansCov (*class in dmae.initializers*), 6
InitPlusPlus (*class in dmae.initializers*), 6

K

kullback_leibler() (*in module dmae.dissimilarities*), 4
kullback_leibler_loss() (*in module dmae.losses*), 17

M

mahalanobis() (*in module dmae.dissimilarities*), 4
mahalanobis_loss() (*in module dmae.losses*), 18
manhattan() (*in module dmae.dissimilarities*), 4
manhattan_loss() (*in module dmae.losses*), 18
minkowsky() (*in module dmae.dissimilarities*), 4
minkowsky_loss() (*in module dmae.losses*), 18
module
    dmae, 20
    dmae.dissimilarities, 3
    dmae.initializers, 5
    dmae.layers, 7
    dmae.losses, 17
    dmae.metrics, 19

T

toroidal_euclidean() (*in module dmae.dissimilarities*), 5
toroidal_euclidean_loss() (*in module dmae.losses*), 19

U

unsupervised_classification_accuracy()
(*in module dmae.metrics*), 19

Z

zero_norm() (*in module dmae.metrics*), 19