ncem Documentation

Release 0.1.4

Anna Schaar

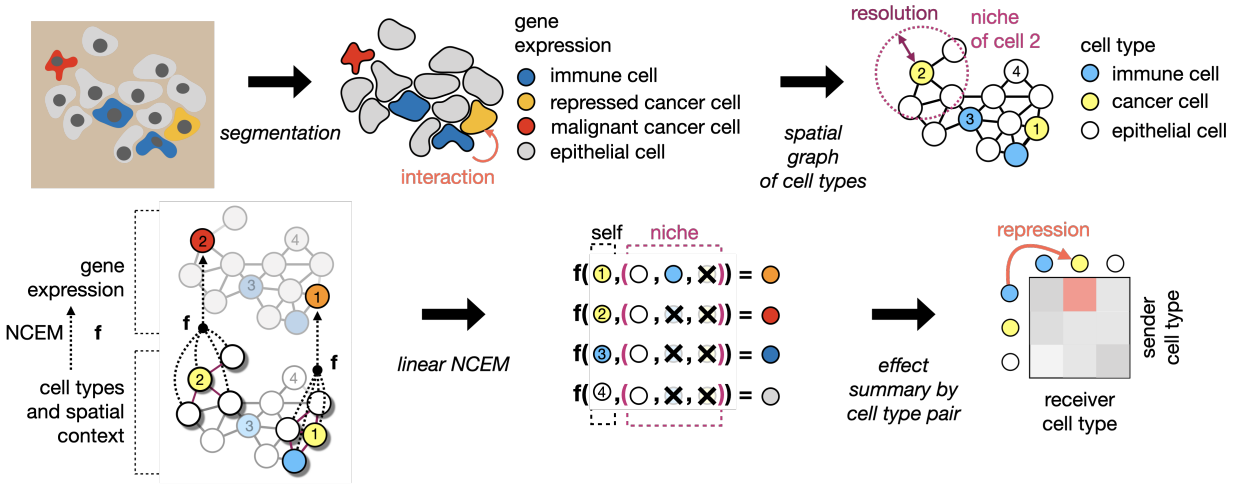
Aug 08, 2022

CONTENTS:

CHAPTER

ONE

NCEM



1.1 Features

ncem is a model repository in a single python package for the manuscript *Fischer, D. S., Schaar, A. C. and Theis, F. Learning cell communication from spatial graphs of cells. 2021.* (preprint)

1.2 Installation

You can install *ncem* via pip from PyPI:

```
$ pip install ncem
```

1.3 Credits

This package was created with cookietemple using Cookiecutter based on Hypermodern_Python_Cookiecutter.

CHAPTER

TWO

INSTALLATION

2.1 Stable release

To install ncem, run this command in your terminal:

```
$ pip install ncem
```

This is the preferred method to install ncem, as it will always install the most recent stable release.

If you don't have pip installed, this Python installation guide can guide you through the process.

2.2 From sources

The sources for ncem can be downloaded from the Github repo. Please note that you require poetry to be installed.

You can either clone the public repository:

```
$ git clone git://github.com/theislab/ncem
```

Or download the tarball:

```
$ curl -OJL https://github.com/theislab/ncem/tarball/main
```

Once you have a copy of the source, you can install it with:

```
$ make install
```

CHAPTER

THREE

API

Import ncem as:

```
import ncem
```

3.1 Estimator classes: *estimators*

Estimator classes from ncem for advanced use.

estimators.Estimator()	Estimator class for models.
estimators.EstimatorGraph()	EstimatorGraph class for spatial models.
estimators.EstimatorNoGraph()	EstimatorNoGraph class for baseline models.
estimators.EstimatorCVAE([use_type_cond, ...])	Estimator class for conditional variational autoencoder models.
estimators.EstimatorCVAEncem([cond_type, ...])	Estimator class for conditional variational autoencoder NCEM models.
estimators.EstimatorED([use_type_cond, ...])	Estimator class for encoder-decoder models.
estimators.EstimatorEDncem([cond_type, ...])	Estimator class for encoder-decoder NCEM models.
estimators.EstimatorInteractions([log_transform])	Estimator class for interactions models.
estimators.EstimatorLinear([log_transform])	Estimator class for linear models.

3.1.1 ncem.estimators.Estimator

class ncem.estimators.**Estimator**

Estimator class for models.

Contains all necessary methods for data loading, model initialization, training, evaluation and prediction.

Attributes

img_keys_all	Return all image keys.
nodes_idx_all	Return all node indices.
nodes_idx_eval	alias of Dict[str, list]
nodes_idx_test	alias of Dict[str, list]
nodes_idx_train	alias of Dict[str, list]
patient_ids_bytarget	Return patient identifiers by target.

continues on next page

Table 1 – continued from previous page

patient_ids_unique	Return unique patient identifiers.
img_to_patient_dict	
complete_img_keys	
a	
h_0	
h_1	
size_factors	
graph_covar	
node_covar	
domains	
covar_selection	
node_types	
node_type_names	
graph_covar_names	
node_feature_names	
n_features_type	
n_features_standard	
n_features_0	
n_features_1	
n_graph_covariates	
n_node_covariates	
n_domains	
max_nodes	
n_eval_nodes_per_graph	
vi_model	
log_transform	

continues on next page

Table 1 – continued from previous page

model_type
adj_type
cond_type
cond_depth
output_layer
steps_per_epoch
validation_steps

ncem.estimators.Estimator.img_keys_all

property Estimator.**img_keys_all**

Return all image keys.

Return type
img_keys_all

ncem.estimators.Estimator.nodes_idx_all

property Estimator.**nodes_idx_all**

Return all node indices.

Return type
nodes_idx_all

ncem.estimators.Estimator.nodes_idx_eval

Estimator.**nodes_idx_eval**

alias of Dict[str, list]

alias of Dict[str, list] .. autoattribute:: Estimator.nodes_idx_eval

ncem.estimators.Estimator.nodes_idx_test

Estimator.**nodes_idx_test**

alias of Dict[str, list]

alias of Dict[str, list] .. autoattribute:: Estimator.nodes_idx_test

ncem.estimators.Estimator.nodes_idx_train

Estimator.**nodes_idx_train**

alias of Dict[str, list]

alias of Dict[str, list] .. autoattribute:: Estimator.nodes_idx_train

ncem.estimators.Estimator.patient_ids_bytarget

property Estimator.**patient_ids_bytarget: ndarray**

Return patient identifiers by target.

Return type
patient_ids_bytarget

ncem.estimators.Estimator.patient_ids_unique

property Estimator.**patient_ids_unique: ndarray**

Return unique patient identifiers.

Return type
patient_ids_unique

ncem.estimators.Estimator.img_to_patient_dict

Estimator.**img_to_patient_dict: Dict[str, str]**

ncem.estimators.Estimator.complete_img_keys

Estimator.**complete_img_keys: List[str]**

ncem.estimators.Estimator.a

Estimator.**a: dict**

ncem.estimators.Estimator.h_0

Estimator.**h_0: Dict[str, ndarray]**

ncem.estimators.Estimator.h_1

Estimator.**h_1: Dict[str, ndarray]**

ncem.estimators.Estimator.size_factors

Estimator.**size_factors**: Dict[str, ndarray]

ncem.estimators.Estimator.graph_covar

Estimator.**graph_covar**: Dict[str, ndarray]

ncem.estimators.Estimator.node_covar

Estimator.**node_covar**: Dict[str, ndarray]

ncem.estimators.Estimator.domains

Estimator.**domains**: Dict[str, ndarray]

ncem.estimators.Estimator.covar_selection

Estimator.**covar_selection**: Optional[Union[List[str], Tuple[str]]]

ncem.estimators.Estimator.node_types

Estimator.**node_types**: Dict[str, ndarray]

ncem.estimators.Estimator.node_type_names

Estimator.**node_type_names**: Dict[str, str]

ncem.estimators.Estimator.graph_covar_names

Estimator.**graph_covar_names**: Dict[str, List[str]]

ncem.estimators.Estimator.node_feature_names

Estimator.**node_feature_names**: List[str]

ncem.estimators.Estimator.n_features_type

Estimator.**n_features_type**: int

ncem.estimators.Estimator.n_features_standard

Estimator.**n_features_standard**: **int**

ncem.estimators.Estimator.n_features_0

Estimator.**n_features_0**: **int**

ncem.estimators.Estimator.n_features_1

Estimator.**n_features_1**: **int**

ncem.estimators.Estimator.n_graph_covariates

Estimator.**n_graph_covariates**: **int**

ncem.estimators.Estimator.n_node_covariates

Estimator.**n_node_covariates**: **int**

ncem.estimators.Estimator.n_domains

Estimator.**n_domains**: **int**

ncem.estimators.Estimator.max_nodes

Estimator.**max_nodes**: **int**

ncem.estimators.Estimator.n_eval_nodes_per_graph

Estimator.**n_eval_nodes_per_graph**: **int**

ncem.estimators.Estimator.vi_model

Estimator.**vi_model**: **bool**

ncem.estimators.Estimator.log_transform

Estimator.**log_transform**: **bool**

ncem.estimators.Estimator.model_type

Estimator.**model_type: str**

ncem.estimators.Estimator.adj_type

Estimator.**adj_type: str**

ncem.estimators.Estimator.cond_type

Estimator.**cond_type: str**

ncem.estimators.Estimator.cond_depth

Estimator.**cond_depth: int**

ncem.estimators.Estimator.output_layer

Estimator.**output_layer: str**

ncem.estimators.Estimator.steps_per_epoch

Estimator.**steps_per_epoch: int**

ncem.estimators.Estimator.validation_steps

Estimator.**validation_steps: int**

Methods

evaluate_any(img_keys, node_idx[, batch_size])	Evaluate model on any given data set.
evaluate_per_node_type([batch_size])	Evaluate model for each node type seperately.
get_data(data_origin, data_path, radius[, ...])	Get data used in estimator classes.
init_model(**kwargs)	Initialize and compiles the model.
predict([batch_size])	Return observed labels and full predictions (including scale model) grouped exactly as in nodes_idx_test.
pretrain_decoder([decoder_epochs, patience, ...])	Pre-train decoder model.
split_data_given(img_keys_test, ...)	Split data by given partition.
split_data_node(test_split, validation_split)	Split nodes randomly into partitions.
split_data_target_cell(target_cell, ...[, seed])	Split nodes randomly into partitions.
train([epochs, epochs_warmup, ...])	Train model.
train_aggressive([aggressive_enc_patience, ...])	Train model aggressive.
train_normal([epochs, patience, ...])	Train model normal.

ncem.estimators.Estimator.evaluate_any

Estimator.**evaluate_any**(*img_keys*, *node_idx*, *batch_size=1*)

Evaluate model on any given data set.

Parameters

- **img_keys** – Image keys.
- **node_idx** – Nodes indices.
- **batch_size** (*int*) – Number of samples. If unspecified, it will default to 1.

Return type
eval_dict

ncem.estimators.Estimator.evaluate_per_node_type

Estimator.**evaluate_per_node_type**(*batch_size=1*)

Evaluate model for each node type seperately.

Parameters
batch_size (*int*) – Number of samples. If unspecified, it will default to 1.

Return type
split_per_node_type, evaluation_per_node_type

ncem.estimators.Estimator.get_data

Estimator.**get_data**(*data_origin*, *data_path*, *radius*, *n_rings=1*, *graph_covar_selection=None*, *node_label_space_id='type'*, *node_feature_space_id='standard'*, *use_covar_node_position=False*, *use_covar_node_label=False*, *use_covar_graph_covar=False*, *domain_type='image'*, *robustness=None*, *robustness_seed=1*, *n_top_genes=None*, *segmentation_robustness=None*, *resimulate_nodes=False*, *resimulate_nodes_w_depdency=False*, *resimulate_nodes_sparsity_rate=0.5*)

Get data used in estimator classes.

Parameters

- **data_origin** (*str*) – Data origin.
- **data_path** (*str*) – Data path.
- **radius** (*int, optional*) – Radius.
- **n_rings** (*int*) – Number of rings of neighbors for grid data.
- **graph_covar_selection** (*list, tuple, optional*) – Selected graph covariates.
- **node_label_space_id** (*str*) – Node label space id.
- **node_feature_space_id** (*str*) – Node feature space id.
- **use_covar_node_position** (*bool*) – Whether to use node position as covariate.
- **use_covar_node_label** (*bool*) – Whether to use node label as covariate.
- **use_covar_graph_covar** (*bool*) – Whether to use graph covariates.
- **domain_type** (*str*) – Covariate that is used as domain.

- **robustness** (*float, optional*) – Optional fraction of images for robustness test.
- **robustness_seed** (*int*) – Seed for robustness analysis
- **n_top_genes** (*int, optional*) – N top genes for highly variable gene selection.
- **segmentation_robustness** (*list, optional*) – Parameters for segmentation robustness fit, float for fraction of nodes and float for signal overflow.
- **resimulate_nodes** (*bool*) –
- **resimulate_nodes_w_depdency** (*bool*) –
- **resimulate_nodes_sparsity_rate** (*float*) –

Raises

ValueError – If sub-selected covar_selection could not be found, *node_label_space_id* or *node_feature_space_id* not recognized

ncem.estimators.Estimator.init_model

abstract Estimator.**init_model**(***kwargs*)

Initialize and compiles the model.

Parameters

kwargs – Arbitrary keyword arguments.

ncem.estimators.Estimator.predict

Estimator.**predict**(*batch_size=1*)

Return observed labels and full predictions (including scale model) grouped exactly as in nodes_idx_test.

Parameters

batch_size (*int*) – Number of samples. If unspecified, it will default to 1.

Return type

predict

ncem.estimators.Estimator.pretrain_decoder

Estimator.**pretrain_decoder**(*decoder_epochs=1000*, *patience=20*, *lr_schedule_min_lr=1e-05*, *lr_schedule_factor=0.2*, *lr_schedule_patience=5*, *initial_epoch=0*, *monitor_partition='val'*, *monitor_metric='loss'*, *log_dir=None*, *callbacks=None*, *early_stopping=True*, *reduce_lr_plateau=True*, ***kwargs*)

Pre-train decoder model.

Use validation loss and maximum number of epochs as termination criteria.

Parameters

- **patience** (*int*) – Number of epochs with no improvement. If unspecified, it will default to 20.
- **lr_schedule_min_lr** (*float*) – Lower bound on the learning rate. If unspecified, it will default to 1e-5.
- **lr_schedule_factor** (*float*) – Factor by which the learning rate will be reduced. new_lr = lr * factor. If unspecified, it will default to 0.2.

- **lr_schedule_patience** (*int*) – Number of epochs with no improvement after which learning rate will be reduced. If unspecified, it will default to 5.
- **initial_epoch** (*int*) – Epoch at which to start training (useful for resuming a previous training run). If unspecified, it will default to 0.
- **monitor_partition** (*str*) – Monitor partition.
- **monitor_metric** (*str*) – Monitor metric.
- **log_dir** (*str, optional*) – Logging directory.
- **callbacks** (*list, optional*) – List of callbacks to be called during training.
- **early_stopping** (*bool*) – Whether to activate early stopping.
- **reduce_lr_plateau** (*bool*) – Whether to reduce learning rate on plateau.
- **decoder_epochs** (*int*) – Integer number of times to iterate over the training data arrays in decoder pretraining. If unspecified, it will default to 1000.
- **kwargs** – Arbitrary keyword arguments.

ncem.estimators.Estimator.split_data_given

Estimator.**split_data_given**(*img_keys_test*, *img_keys_train*, *img_keys_eval*, *nodes_idx_test*, *nodes_idx_train*, *nodes_idx_eval*)

Split data by given partition.

Parameters

- **img_keys_test** – Test image keys.
- **img_keys_train** – Train image keys.
- **img_keys_eval** – Evaluation image keys.
- **nodes_idx_test** – Test node indices.
- **nodes_idx_train** – Train node indices.
- **nodes_idx_eval** – Evaluation node indices.

ncem.estimators.Estimator.split_data_node

Estimator.**split_data_node**(*test_split*, *validation_split*, *seed=1*)

Split nodes randomly into partitions.

Parameters

- **test_split** (*float*) – Fraction of total nodes to be in test set.
- **validation_split** (*float*) – Fraction of train-eval nodes to be in validation split.
- **seed** (*int*) – Seed for random selection of observations.

Raises

ValueError – If evaluation or test dataset are empty.

ncem.estimators.Estimator.split_data_target_cell

Estimator.**split_data_target_cell**(*target_cell*, *test_split*, *validation_split*, *seed=1*)

Split nodes randomly into partitions.

Parameters

- **target_cell** (*str*) – Target cell type.
- **test_split** (*float*) – Fraction of total nodes to be in test set.
- **validation_split** (*float*) – Fraction of train-eval nodes to be in validation split.
- **seed** (*int*) – Seed for random selection of observations.

Raises

ValueError – If evaluation or test dataset are empty.

ncem.estimators.Estimator.train

Estimator.**train**(*epochs=1000*, *epochs_warmup=0*, *max_steps_per_epoch=20*, *batch_size=16*, *validation_batch_size=16*, *max_validation_steps=10*, *shuffle_buffer_size=10000*, *patience=20*, *lr_schedule_min_lr=1e-05*, *lr_schedule_factor=0.2*, *lr_schedule_patience=5*, *initial_epoch=0*, *monitor_partition='val'*, *monitor_metric='loss'*, *log_dir=None*, *callbacks=None*, *early_stopping=True*, *reduce_lr_plateau=True*, *pretrain_decoder=False*, *decoder_epochs=1000*, *decoder_patience=20*, *decoder_callbacks=None*, *aggressive=False*, *aggressive_enc_patience=10*, *aggressive_epochs=5*, *seed=1234*, ***kwargs*)

Train model.

Use validation loss and maximum number of epochs as termination criteria.

Parameters

- **epochs** (*int*) – Integer number of times to iterate over the training data arrays. If unspecified, it will default to 1000.
- **epochs_warmup** (*int*) – Integer number of times to iterate over the training data arrays in warm up (without early stopping). If unspecified, it will default to 0.
- **max_steps_per_epoch** (*int, optional*) – Maximal steps per epoch. If unspecified, it will default to 20.
- **batch_size** (*int*) – Number of samples per gradient update. If unspecified, it will default to 16.
- **validation_batch_size** (*int*) – Number of samples in validation. If unspecified, it will default to 16.
- **max_validation_steps** (*int*) – Maximal steps per validation. If unspecified, it will default to 10.
- **shuffle_buffer_size** (*int, optional*) – Shuffle buffer size. If unspecified, it will default to 1e4.
- **patience** (*int*) – Number of epochs with no improvement. If unspecified, it will default to 20.
- **lr_schedule_min_lr** (*float*) – Lower bound on the learning rate. If unspecified, it will default to 1e-5.

- **lr_schedule_factor** (*float*) – Factor by which the learning rate will be reduced. new_lr = lr * factor. If unspecified, it will default to 0.2.
- **lr_schedule_patience** (*int*) – Number of epochs with no improvement after which learning rate will be reduced. If unspecified, it will default to 5.
- **initial_epoch** (*int*) – Epoch at which to start training (useful for resuming a previous training run). If unspecified, it will default to 0.
- **monitor_partition** (*str*) – Monitor partition.
- **monitor_metric** (*str*) – Monitor metric.
- **log_dir** (*str, optional*) – Logging directory.
- **callbacks** (*list, optional*) – List of callbacks to be called during training.
- **early_stopping** (*bool*) – Whether to activate early stopping.
- **reduce_lr_plateau** (*bool*) – Whether to reduce learning rate on plateau.
- **pretrain_decoder** (*bool*) – Whether to pretrain the decoder model.
- **decoder_epochs** (*int*) – Integer number of times to iterate over the training data arrays in decoder pretraining. If unspecified, it will default to 1000.
- **decoder_patience** (*int*) – Number of epochs with no improvement in decoder pretraining. If unspecified, it will default to 20.
- **decoder_callbacks** (*list, optional*) – List of callbacks to be called during decoder pretraining.
- **aggressive** (*bool*) – Whether to train aggressive.
- **aggressive_enc_patience** (*int*) – Number of epochs with no improvement in aggressive training. If unspecified, it will default to 10.
- **aggressive_epochs** (*int*) – Integer number of times to iterate over the training data arrays in aggressive training. If unspecified, it will default to 5.
- **seed** (*int*) – Random seed for reproduability.
- **kwargs** – Arbitrary keyword arguments.

ncem.estimators.Estimator.train_aggressive

Estimator.**train_aggressive**(*aggressive_enc_patience=10*, *aggressive_epochs=5*)

Train model aggressive.

Parameters

- **aggressive_enc_patience** (*int*) – Number of epochs with no improvement in aggressive training. If unspecified, it will default to 10.
- **aggressive_epochs** (*int*) – Integer number of times to iterate over the training data arrays in aggressive training. If unspecified, it will default to 5.

ncem.estimators.Estimator.train_normal

Estimator.**train_normal**(*epochs=1000*, *patience=20*, *lr_schedule_min_lr=1e-05*, *lr_schedule_factor=0.2*, *lr_schedule_patience=5*, *initial_epoch=0*, *monitor_partition='val'*, *monitor_metric='loss'*, *log_dir=None*, *callbacks=None*, *early_stopping=True*, *reduce_lr_plateau=True*, ***kwargs*)

Train model normal.

Use validation loss and maximum number of epochs as termination criteria.

Parameters

- **epochs** (*int*) – Integer number of times to iterate over the training data arrays. If unspecified, it will default to 1000.
- **patience** (*int*) – Number of epochs with no improvement. If unspecified, it will default to 20.
- **lr_schedule_min_lr** (*float*) – Lower bound on the learning rate. If unspecified, it will default to 1e-5.
- **lr_schedule_factor** (*float*) – Factor by which the learning rate will be reduced. new_lr = lr * factor. If unspecified, it will default to 0.2.
- **lr_schedule_patience** (*int*) – Number of epochs with no improvement after which learning rate will be reduced. If unspecified, it will default to 5.
- **initial_epoch** (*int*) – Epoch at which to start training (useful for resuming a previous training run). If unspecified, it will default to 0.
- **monitor_partition** (*str*) – Monitor partition.
- **monitor_metric** (*str*) – Monitor metric.
- **log_dir** (*str, optional*) – Logging directory.
- **callbacks** (*list, optional*) – List of callbacks to be called during training.
- **early_stopping** (*bool*) – Whether to activate early stopping.
- **reduce_lr_plateau** (*bool*) – Whether to reduce learning rate on plateau.
- **kwargs** – Arbitrary keyword arguments.

3.1.2 ncem.estimators.EstimatorGraph

class ncem.estimators.**EstimatorGraph**

EstimatorGraph class for spatial models.

Attributes

img_keys_all	Return all image keys.
nodes_idx_all	Return all node indices.
nodes_idx_eval	alias of Dict[str, list]
nodes_idx_test	alias of Dict[str, list]
nodes_idx_train	alias of Dict[str, list]
patient_ids_bytarget	Return patient identifiers by target.
patient_ids_unique	Return unique patient identifiers.

ncem.estimators.EstimatorGraph.img_keys_all

property EstimatorGraph.**img_keys_all**

Return all image keys.

Return type
img_keys_all

ncem.estimators.EstimatorGraph.nodes_idx_all

property EstimatorGraph.**nodes_idx_all**

Return all node indices.

Return type
nodes_idx_all

ncem.estimators.EstimatorGraph.nodes_idx_eval

EstimatorGraph.**nodes_idx_eval**

alias of Dict[str, list]

alias of Dict[str, list] .. autoattribute:: EstimatorGraph.nodes_idx_eval

ncem.estimators.EstimatorGraph.nodes_idx_test

EstimatorGraph.**nodes_idx_test**

alias of Dict[str, list]

alias of Dict[str, list] .. autoattribute:: EstimatorGraph.nodes_idx_test

ncem.estimators.EstimatorGraph.nodes_idx_train

EstimatorGraph.**nodes_idx_train**

alias of Dict[str, list]

alias of Dict[str, list] .. autoattribute:: EstimatorGraph.nodes_idx_train

ncem.estimators.EstimatorGraph.patient_ids_bytarget

property EstimatorGraph.**patient_ids_bytarget: ndarray**

Return patient identifiers by target.

Return type
patient_ids_bytarget

ncem.estimators.EstimatorGraph.patient_ids_unique

property EstimatorGraph.**patient_ids_unique: ndarray**

Return unique patient identifiers.

Return type
patient_ids_unique

Methods

evaluate_any(img_keys, node_idx[, batch_size])	Evaluate model on any given data set.
evaluate_per_node_type([batch_size])	Evaluate model for each node type seperately.
get_data(data_origin, data_path, radius[, ...])	Get data used in estimator classes.
init_model(**kwargs)	Initialize EstimatorGraph.
predict([batch_size])	Return observed labels and full predictions (including scale model) grouped exactly as in nodes_idx_test.
pretrain_decoder([decoder_epochs, patience, ...])	Pre-train decoder model.
split_data_given(img_keys_test, ...)	Split data by given partition.
split_data_node(test_split, validation_split)	Split nodes randomly into partitions.
split_data_target_cell(target_cell, ...[, seed])	Split nodes randomly into partitions.
train([epochs, epochs_warmup, ...])	Train model.
train_aggressive([aggressive_enc_patience, ...])	Train model aggressive.
train_normal([epochs, patience, ...])	Train model normal.

ncem.estimators.EstimatorGraph.evaluate_any

EstimatorGraph.**evaluate_any**(*img_keys*, *node_idx*, *batch_size=1*)

Evaluate model on any given data set.

Parameters

- **img_keys** – Image keys.
- **node_idx** – Nodes indices.
- **batch_size** (*int*) – Number of samples. If unspecified, it will default to 1.

Return type
eval_dict

ncem.estimators.EstimatorGraph.evaluate_per_node_type

EstimatorGraph.**evaluate_per_node_type**(*batch_size=1*)

Evaluate model for each node type seperately.

Parameters
batch_size (*int*) – Number of samples. If unspecified, it will default to 1.

Return type
split_per_node_type, evaluation_per_node_type

ncem.estimators.EstimatorGraph.get_data

EstimatorGraph.**get_data**(*data_origin*, *data_path*, *radius*, *n_rings=1*, *graph_covar_selection=None*, *node_label_space_id='type'*, *node_feature_space_id='standard'*, *use_covar_node_position=False*, *use_covar_node_label=False*, *use_covar_graph_covar=False*, *domain_type='image'*, *robustness=None*, *robustness_seed=1*, *n_top_genes=None*, *segmentation_robustness=None*, *resimulate_nodes=False*, *resimulate_nodes_w_depdency=False*, *resimulate_nodes_sparsity_rate=0.5*)

Get data used in estimator classes.

Parameters

- **data_origin** (*str*) – Data origin.
- **data_path** (*str*) – Data path.
- **radius** (*int, optional*) – Radius.
- **n_rings** (*int*) – Number of rings of neighbors for grid data.
- **graph_covar_selection** (*list, tuple, optional*) – Selected graph covariates.
- **node_label_space_id** (*str*) – Node label space id.
- **node_feature_space_id** (*str*) – Node feature space id.
- **use_covar_node_position** (*bool*) – Whether to use node position as covariate.
- **use_covar_node_label** (*bool*) – Whether to use node label as covariate.
- **use_covar_graph_covar** (*bool*) – Whether to use graph covariates.
- **domain_type** (*str*) – Covariate that is used as domain.
- **robustness** (*float, optional*) – Optional fraction of images for robustness test.
- **robustness_seed** (*int*) – Seed for robustness analysis
- **n_top_genes** (*int, optional*) – N top genes for highly variable gene selection.
- **segmentation_robustness** (*list, optional*) – Parameters for segmentation robustness fit, float for fraction of nodes and float for signal overflow.
- **resimulate_nodes** (*bool*) –
- **resimulate_nodes_w_depdency** (*bool*) –
- **resimulate_nodes_sparsity_rate** (*float*) –

Raises

ValueError – If sub-selected covar_selection could not be found, *node_label_space_id* or *node_feature_space_id* not recognized

ncem.estimators.EstimatorGraph.init_model

EstimatorGraph.**init_model**(***kwargs*)

Initialize EstimatorGraph.

Parameters
kwargs – Arbitrary keyword arguments.

ncem.estimators.EstimatorGraph.predict

EstimatorGraph.**predict**(*batch_size=1*)

Return observed labels and full predictions (including scale model) grouped exactly as in nodes_idx_test.

Parameters
batch_size (*int*) – Number of samples. If unspecified, it will default to 1.

Return type
predict

ncem.estimators.EstimatorGraph.pretrain_decoder

EstimatorGraph.**pretrain_decoder**(*decoder_epochs=1000*, *patience=20*, *lr_schedule_min_lr=1e-05*, *lr_schedule_factor=0.2*, *lr_schedule_patience=5*, *initial_epoch=0*, *monitor_partition='val'*, *monitor_metric='loss'*, *log_dir=None*, *callbacks=None*, *early_stopping=True*, *reduce_lr_plateau=True*, ***kwargs*)

Pre-train decoder model.

Use validation loss and maximum number of epochs as termination criteria.

Parameters

- **patience** (*int*) – Number of epochs with no improvement. If unspecified, it will default to 20.
- **lr_schedule_min_lr** (*float*) – Lower bound on the learning rate. If unspecified, it will default to 1e-5.
- **lr_schedule_factor** (*float*) – Factor by which the learning rate will be reduced. new_lr = lr * factor. If unspecified, it will default to 0.2.
- **lr_schedule_patience** (*int*) – Number of epochs with no improvement after which learning rate will be reduced. If unspecified, it will default to 5.
- **initial_epoch** (*int*) – Epoch at which to start training (useful for resuming a previous training run). If unspecified, it will default to 0.
- **monitor_partition** (*str*) – Monitor partition.
- **monitor_metric** (*str*) – Monitor metric.
- **log_dir** (*str, optional*) – Logging directory.
- **callbacks** (*list, optional*) – List of callbacks to be called during training.
- **early_stopping** (*bool*) – Whether to activate early stopping.
- **reduce_lr_plateau** (*bool*) – Whether to reduce learning rate on plateau.

- **decoder_epochs** (*int*) – Integer number of times to iterate over the training data arrays in decoder pretraining. If unspecified, it will default to 1000.
- **kwargs** – Arbitrary keyword arguments.

ncem.estimators.EstimatorGraph.split_data_given

EstimatorGraph.**split_data_given**(*img_keys_test*, *img_keys_train*, *img_keys_eval*, *nodes_idx_test*, *nodes_idx_train*, *nodes_idx_eval*)

Split data by given partition.

Parameters

- **img_keys_test** – Test image keys.
- **img_keys_train** – Train image keys.
- **img_keys_eval** – Evaluation image keys.
- **nodes_idx_test** – Test node indices.
- **nodes_idx_train** – Train node indices.
- **nodes_idx_eval** – Evaluation node indices.

ncem.estimators.EstimatorGraph.split_data_node

EstimatorGraph.**split_data_node**(*test_split*, *validation_split*, *seed=1*)

Split nodes randomly into partitions.

Parameters

- **test_split** (*float*) – Fraction of total nodes to be in test set.
- **validation_split** (*float*) – Fraction of train-eval nodes to be in validation split.
- **seed** (*int*) – Seed for random selection of observations.

Raises

ValueError – If evaluation or test dataset are empty.

ncem.estimators.EstimatorGraph.split_data_target_cell

EstimatorGraph.**split_data_target_cell**(*target_cell*, *test_split*, *validation_split*, *seed=1*)

Split nodes randomly into partitions.

Parameters

- **target_cell** (*str*) – Target cell type.
- **test_split** (*float*) – Fraction of total nodes to be in test set.
- **validation_split** (*float*) – Fraction of train-eval nodes to be in validation split.
- **seed** (*int*) – Seed for random selection of observations.

Raises

ValueError – If evaluation or test dataset are empty.

ncem.estimators.EstimatorGraph.train

EstimatorGraph.**train**(*epochs=1000*, *epochs_warmup=0*, *max_steps_per_epoch=20*, *batch_size=16*, *validation_batch_size=16*, *max_validation_steps=10*, *shuffle_buffer_size=10000*, *patience=20*, *lr_schedule_min_lr=1e-05*, *lr_schedule_factor=0.2*, *lr_schedule_patience=5*, *initial_epoch=0*, *monitor_partition='val'*, *monitor_metric='loss'*, *log_dir=None*, *callbacks=None*, *early_stopping=True*, *reduce_lr_plateau=True*, *pretrain_decoder=False*, *decoder_epochs=1000*, *decoder_patience=20*, *decoder_callbacks=None*, *aggressive=False*, *aggressive_enc_patience=10*, *aggressive_epochs=5*, *seed=1234*, ***kwargs*)

Train model.

Use validation loss and maximum number of epochs as termination criteria.

Parameters

- **epochs** (*int*) – Integer number of times to iterate over the training data arrays. If unspecified, it will default to 1000.
- **epochs_warmup** (*int*) – Integer number of times to iterate over the training data arrays in warm up (without early stopping). If unspecified, it will default to 0.
- **max_steps_per_epoch** (*int, optional*) – Maximal steps per epoch. If unspecified, it will default to 20.
- **batch_size** (*int*) – Number of samples per gradient update. If unspecified, it will default to 16.
- **validation_batch_size** (*int*) – Number of samples in validation. If unspecified, it will default to 16.
- **max_validation_steps** (*int*) – Maximal steps per validation. If unspecified, it will default to 10.
- **shuffle_buffer_size** (*int, optional*) – Shuffle buffer size. If unspecified, it will default to 1e4.
- **patience** (*int*) – Number of epochs with no improvement. If unspecified, it will default to 20.
- **lr_schedule_min_lr** (*float*) – Lower bound on the learning rate. If unspecified, it will default to 1e-5.
- **lr_schedule_factor** (*float*) – Factor by which the learning rate will be reduced. new_lr = lr * factor. If unspecified, it will default to 0.2.
- **lr_schedule_patience** (*int*) – Number of epochs with no improvement after which learning rate will be reduced. If unspecified, it will default to 5.
- **initial_epoch** (*int*) – Epoch at which to start training (useful for resuming a previous training run). If unspecified, it will default to 0.
- **monitor_partition** (*str*) – Monitor partition.
- **monitor_metric** (*str*) – Monitor metric.
- **log_dir** (*str, optional*) – Logging directory.
- **callbacks** (*list, optional*) – List of callbacks to be called during training.
- **early_stopping** (*bool*) – Whether to activate early stopping.
- **reduce_lr_plateau** (*bool*) – Whether to reduce learning rate on plateau.

- **pretrain_decoder** (*bool*) – Whether to pretrain the decoder model.
- **decoder_epochs** (*int*) – Integer number of times to iterate over the training data arrays in decoder pretraining. If unspecified, it will default to 1000.
- **decoder_patience** (*int*) – Number of epochs with no improvement in decoder pretraining. If unspecified, it will default to 20.
- **decoder_callbacks** (*list, optional*) – List of callbacks to be called during decoder pretraining.
- **aggressive** (*bool*) – Whether to train aggressive.
- **aggressive_enc_patience** (*int*) – Number of epochs with no improvement in aggressive training. If unspecified, it will default to 10.
- **aggressive_epochs** (*int*) – Integer number of times to iterate over the training data arrays in aggressive training. If unspecified, it will default to 5.
- **seed** (*int*) – Random seed for reproduability.
- **kwargs** – Arbitrary keyword arguments.

ncem.estimators.EstimatorGraph.train_aggressive

EstimatorGraph.**train_aggressive**(*aggressive_enc_patience=10*, *aggressive_epochs=5*)

Train model aggressive.

Parameters

- **aggressive_enc_patience** (*int*) – Number of epochs with no improvement in aggressive training. If unspecified, it will default to 10.
- **aggressive_epochs** (*int*) – Integer number of times to iterate over the training data arrays in aggressive training. If unspecified, it will default to 5.

ncem.estimators.EstimatorGraph.train_normal

EstimatorGraph.**train_normal**(*epochs=1000*, *patience=20*, *lr_schedule_min_lr=1e-05*, *lr_schedule_factor=0.2*, *lr_schedule_patience=5*, *initial_epoch=0*, *monitor_partition='val'*, *monitor_metric='loss'*, *log_dir=None*, *callbacks=None*, *early_stopping=True*, *reduce_lr_plateau=True*, ***kwargs*)

Train model normal.

Use validation loss and maximum number of epochs as termination criteria.

Parameters

- **epochs** (*int*) – Integer number of times to iterate over the training data arrays. If unspecified, it will default to 1000.
- **patience** (*int*) – Number of epochs with no improvement. If unspecified, it will default to 20.
- **lr_schedule_min_lr** (*float*) – Lower bound on the learning rate. If unspecified, it will default to 1e-5.
- **lr_schedule_factor** (*float*) – Factor by which the learning rate will be reduced. new_lr = lr * factor. If unspecified, it will default to 0.2.

- **lr_schedule_patience** (*int*) – Number of epochs with no improvement after which learning rate will be reduced. If unspecified, it will default to 5.
- **initial_epoch** (*int*) – Epoch at which to start training (useful for resuming a previous training run). If unspecified, it will default to 0.
- **monitor_partition** (*str*) – Monitor partition.
- **monitor_metric** (*str*) – Monitor metric.
- **log_dir** (*str, optional*) – Logging directory.
- **callbacks** (*list, optional*) – List of callbacks to be called during training.
- **early_stopping** (*bool*) – Whether to activate early stopping.
- **reduce_lr_plateau** (*bool*) – Whether to reduce learning rate on plateau.
- **kwargs** – Arbitrary keyword arguments.

3.1.3 ncem.estimators.EstimatorNoGraph

class ncem.estimators.**EstimatorNoGraph**

EstimatorNoGraph class for baseline models.

Attributes

img_keys_all	Return all image keys.
nodes_idx_all	Return all node indices.
nodes_idx_eval	alias of Dict[str, list]
nodes_idx_test	alias of Dict[str, list]
nodes_idx_train	alias of Dict[str, list]
patient_ids_bytarget	Return patient identifiers by target.
patient_ids_unique	Return unique patient identifiers.

ncem.estimators.EstimatorNoGraph.img_keys_all

property EstimatorNoGraph.**img_keys_all**

Return all image keys.

Return type
img_keys_all

ncem.estimators.EstimatorNoGraph.nodes_idx_all

property EstimatorNoGraph.**nodes_idx_all**

Return all node indices.

Return type
nodes_idx_all

ncem.estimators.EstimatorNoGraph.nodes_idx_eval

EstimatorNoGraph.**nodes_idx_eval**
alias of Dict[str, list]

alias of Dict[str, list] .. autoattribute:: EstimatorNoGraph.nodes_idx_eval

ncem.estimators.EstimatorNoGraph.nodes_idx_test

EstimatorNoGraph.**nodes_idx_test**
alias of Dict[str, list]

alias of Dict[str, list] .. autoattribute:: EstimatorNoGraph.nodes_idx_test

ncem.estimators.EstimatorNoGraph.nodes_idx_train

EstimatorNoGraph.**nodes_idx_train**
alias of Dict[str, list]

alias of Dict[str, list] .. autoattribute:: EstimatorNoGraph.nodes_idx_train

ncem.estimators.EstimatorNoGraph.patient_ids_bytarget

property EstimatorNoGraph.**patient_ids_bytarget: ndarray**
Return patient identifiers by target.

Return type
patient_ids_bytarget

ncem.estimators.EstimatorNoGraph.patient_ids_unique

property EstimatorNoGraph.**patient_ids_unique: ndarray**
Return unique patient identifiers.

Return type
patient_ids_unique

Methods

evaluate_any(img_keys, node_idx[, batch_size])	Evaluate model on any given data set.
evaluate_per_node_type([batch_size])	Evaluate model for each node type seperately.
get_data(data_origin, data_path, radius[, ...])	Get data used in estimator classes.
init_model(**kwargs)	Initialize EstimatorNoGraph.
predict([batch_size])	Return observed labels and full predictions (including scale model) grouped exactly as in nodes_idx_test.
pretrain_decoder([decoder_epochs, patience, ...])	Pre-train decoder model.
split_data_given(img_keys_test, ...)	Split data by given partition.
split_data_node(test_split, validation_split)	Split nodes randomly into partitions.
split_data_target_cell(target_cell, ...[, seed])	Split nodes randomly into partitions.
train([epochs, epochs_warmup, ...])	Train model.
train_aggressive([aggressive_enc_patience, ...])	Train model aggressive.
train_normal([epochs, patience, ...])	Train model normal.

ncem.estimators.EstimatorNoGraph.evaluate_any

EstimatorNoGraph.**evaluate_any**(*img_keys*, *node_idx*, *batch_size=1*)

Evaluate model on any given data set.

Parameters

- **img_keys** – Image keys.
- **node_idx** – Nodes indices.
- **batch_size** (*int*) – Number of samples. If unspecified, it will default to 1.

Return type
eval_dict

ncem.estimators.EstimatorNoGraph.evaluate_per_node_type

EstimatorNoGraph.**evaluate_per_node_type**(*batch_size=1*)

Evaluate model for each node type seperately.

Parameters
batch_size (*int*) – Number of samples. If unspecified, it will default to 1.

Return type
split_per_node_type, evaluation_per_node_type

ncem.estimators.EstimatorNoGraph.get_data

EstimatorNoGraph.**get_data**(*data_origin*, *data_path*, *radius*, *n_rings=1*, *graph_covar_selection=None*, *node_label_space_id='type'*, *node_feature_space_id='standard'*, *use_covar_node_position=False*, *use_covar_node_label=False*, *use_covar_graph_covar=False*, *domain_type='image'*, *robustness=None*, *robustness_seed=1*, *n_top_genes=None*, *segmentation_robustness=None*, *resimulate_nodes=False*, *resimulate_nodes_w_depdency=False*, *resimulate_nodes_sparsity_rate=0.5*)

Get data used in estimator classes.

Parameters

- **data_origin** (*str*) – Data origin.
- **data_path** (*str*) – Data path.
- **radius** (*int, optional*) – Radius.
- **n_rings** (*int*) – Number of rings of neighbors for grid data.
- **graph_covar_selection** (*list, tuple, optional*) – Selected graph covariates.
- **node_label_space_id** (*str*) – Node label space id.
- **node_feature_space_id** (*str*) – Node feature space id.
- **use_covar_node_position** (*bool*) – Whether to use node position as covariate.
- **use_covar_node_label** (*bool*) – Whether to use node label as covariate.
- **use_covar_graph_covar** (*bool*) – Whether to use graph covariates.
- **domain_type** (*str*) – Covariate that is used as domain.
- **robustness** (*float, optional*) – Optional fraction of images for robustness test.
- **robustness_seed** (*int*) – Seed for robustness analysis
- **n_top_genes** (*int, optional*) – N top genes for highly variable gene selection.
- **segmentation_robustness** (*list, optional*) – Parameters for segmentation robustness fit, float for fraction of nodes and float for signal overflow.
- **resimulate_nodes** (*bool*) –
- **resimulate_nodes_w_depdency** (*bool*) –
- **resimulate_nodes_sparsity_rate** (*float*) –

Raises

ValueError – If sub-selected covar_selection could not be found, *node_label_space_id* or *node_feature_space_id* not recognized

ncem.estimators.EstimatorNoGraph.init_model

EstimatorNoGraph.**init_model**(***kwargs*)

Initialize EstimatorNoGraph.

Parameters
kwargs – Arbitrary keyword arguments.

ncem.estimators.EstimatorNoGraph.predict

EstimatorNoGraph.**predict**(*batch_size=1*)

Return observed labels and full predictions (including scale model) grouped exactly as in nodes_idx_test.

Parameters
batch_size (*int*) – Number of samples. If unspecified, it will default to 1.

Return type
predict

ncem.estimators.EstimatorNoGraph.pretrain_decoder

EstimatorNoGraph.**pretrain_decoder**(*decoder_epochs=1000*, *patience=20*, *lr_schedule_min_lr=1e-05*, *lr_schedule_factor=0.2*, *lr_schedule_patience=5*, *initial_epoch=0*, *monitor_partition='val'*, *monitor_metric='loss'*, *log_dir=None*, *callbacks=None*, *early_stopping=True*, *reduce_lr_plateau=True*, ***kwargs*)

Pre-train decoder model.

Use validation loss and maximum number of epochs as termination criteria.

Parameters

- **patience** (*int*) – Number of epochs with no improvement. If unspecified, it will default to 20.
- **lr_schedule_min_lr** (*float*) – Lower bound on the learning rate. If unspecified, it will default to 1e-5.
- **lr_schedule_factor** (*float*) – Factor by which the learning rate will be reduced. new_lr = lr * factor. If unspecified, it will default to 0.2.
- **lr_schedule_patience** (*int*) – Number of epochs with no improvement after which learning rate will be reduced. If unspecified, it will default to 5.
- **initial_epoch** (*int*) – Epoch at which to start training (useful for resuming a previous training run). If unspecified, it will default to 0.
- **monitor_partition** (*str*) – Monitor partition.
- **monitor_metric** (*str*) – Monitor metric.
- **log_dir** (*str, optional*) – Logging directory.
- **callbacks** (*list, optional*) – List of callbacks to be called during training.
- **early_stopping** (*bool*) – Whether to activate early stopping.
- **reduce_lr_plateau** (*bool*) – Whether to reduce learning rate on plateau.

- **decoder_epochs** (*int*) – Integer number of times to iterate over the training data arrays in decoder pretraining. If unspecified, it will default to 1000.
- **kwargs** – Arbitrary keyword arguments.

ncem.estimators.EstimatorNoGraph.split_data_given

EstimatorNoGraph.**split_data_given**(*img_keys_test*, *img_keys_train*, *img_keys_eval*, *nodes_idx_test*, *nodes_idx_train*, *nodes_idx_eval*)

Split data by given partition.

Parameters

- **img_keys_test** – Test image keys.
- **img_keys_train** – Train image keys.
- **img_keys_eval** – Evaluation image keys.
- **nodes_idx_test** – Test node indices.
- **nodes_idx_train** – Train node indices.
- **nodes_idx_eval** – Evaluation node indices.

ncem.estimators.EstimatorNoGraph.split_data_node

EstimatorNoGraph.**split_data_node**(*test_split*, *validation_split*, *seed=1*)

Split nodes randomly into partitions.

Parameters

- **test_split** (*float*) – Fraction of total nodes to be in test set.
- **validation_split** (*float*) – Fraction of train-eval nodes to be in validation split.
- **seed** (*int*) – Seed for random selection of observations.

Raises

ValueError – If evaluation or test dataset are empty.

ncem.estimators.EstimatorNoGraph.split_data_target_cell

EstimatorNoGraph.**split_data_target_cell**(*target_cell*, *test_split*, *validation_split*, *seed=1*)

Split nodes randomly into partitions.

Parameters

- **target_cell** (*str*) – Target cell type.
- **test_split** (*float*) – Fraction of total nodes to be in test set.
- **validation_split** (*float*) – Fraction of train-eval nodes to be in validation split.
- **seed** (*int*) – Seed for random selection of observations.

Raises

ValueError – If evaluation or test dataset are empty.

ncem.estimators.EstimatorNoGraph.train

EstimatorNoGraph.**train**(*epochs=1000*, *epochs_warmup=0*, *max_steps_per_epoch=20*, *batch_size=16*, *validation_batch_size=16*, *max_validation_steps=10*, *shuffle_buffer_size=10000*, *patience=20*, *lr_schedule_min_lr=1e-05*, *lr_schedule_factor=0.2*, *lr_schedule_patience=5*, *initial_epoch=0*, *monitor_partition='val'*, *monitor_metric='loss'*, *log_dir=None*, *callbacks=None*, *early_stopping=True*, *reduce_lr_plateau=True*, *pretrain_decoder=False*, *decoder_epochs=1000*, *decoder_patience=20*, *decoder_callbacks=None*, *aggressive=False*, *aggressive_enc_patience=10*, *aggressive_epochs=5*, *seed=1234*, ***kwargs*)

Train model.

Use validation loss and maximum number of epochs as termination criteria.

Parameters

- **epochs** (*int*) – Integer number of times to iterate over the training data arrays. If unspecified, it will default to 1000.
- **epochs_warmup** (*int*) – Integer number of times to iterate over the training data arrays in warm up (without early stopping). If unspecified, it will default to 0.
- **max_steps_per_epoch** (*int, optional*) – Maximal steps per epoch. If unspecified, it will default to 20.
- **batch_size** (*int*) – Number of samples per gradient update. If unspecified, it will default to 16.
- **validation_batch_size** (*int*) – Number of samples in validation. If unspecified, it will default to 16.
- **max_validation_steps** (*int*) – Maximal steps per validation. If unspecified, it will default to 10.
- **shuffle_buffer_size** (*int, optional*) – Shuffle buffer size. If unspecified, it will default to 1e4.
- **patience** (*int*) – Number of epochs with no improvement. If unspecified, it will default to 20.
- **lr_schedule_min_lr** (*float*) – Lower bound on the learning rate. If unspecified, it will default to 1e-5.
- **lr_schedule_factor** (*float*) – Factor by which the learning rate will be reduced. new_lr = lr * factor. If unspecified, it will default to 0.2.
- **lr_schedule_patience** (*int*) – Number of epochs with no improvement after which learning rate will be reduced. If unspecified, it will default to 5.
- **initial_epoch** (*int*) – Epoch at which to start training (useful for resuming a previous training run). If unspecified, it will default to 0.
- **monitor_partition** (*str*) – Monitor partition.
- **monitor_metric** (*str*) – Monitor metric.
- **log_dir** (*str, optional*) – Logging directory.
- **callbacks** (*list, optional*) – List of callbacks to be called during training.
- **early_stopping** (*bool*) – Whether to activate early stopping.
- **reduce_lr_plateau** (*bool*) – Whether to reduce learning rate on plateau.

- **pretrain_decoder** (*bool*) – Whether to pretrain the decoder model.
- **decoder_epochs** (*int*) – Integer number of times to iterate over the training data arrays in decoder pretraining. If unspecified, it will default to 1000.
- **decoder_patience** (*int*) – Number of epochs with no improvement in decoder pretraining. If unspecified, it will default to 20.
- **decoder_callbacks** (*list, optional*) – List of callbacks to be called during decoder pretraining.
- **aggressive** (*bool*) – Whether to train aggressive.
- **aggressive_enc_patience** (*int*) – Number of epochs with no improvement in aggressive training. If unspecified, it will default to 10.
- **aggressive_epochs** (*int*) – Integer number of times to iterate over the training data arrays in aggressive training. If unspecified, it will default to 5.
- **seed** (*int*) – Random seed for reproduability.
- **kwargs** – Arbitrary keyword arguments.

ncem.estimators.EstimatorNoGraph.train_aggressive

EstimatorNoGraph.**train_aggressive**(*aggressive_enc_patience=10*, *aggressive_epochs=5*)

Train model aggressive.

Parameters

- **aggressive_enc_patience** (*int*) – Number of epochs with no improvement in aggressive training. If unspecified, it will default to 10.
- **aggressive_epochs** (*int*) – Integer number of times to iterate over the training data arrays in aggressive training. If unspecified, it will default to 5.

ncem.estimators.EstimatorNoGraph.train_normal

EstimatorNoGraph.**train_normal**(*epochs=1000*, *patience=20*, *lr_schedule_min_lr=1e-05*, *lr_schedule_factor=0.2*, *lr_schedule_patience=5*, *initial_epoch=0*, *monitor_partition='val'*, *monitor_metric='loss'*, *log_dir=None*, *callbacks=None*, *early_stopping=True*, *reduce_lr_plateau=True*, ***kwargs*)

Train model normal.

Use validation loss and maximum number of epochs as termination criteria.

Parameters

- **epochs** (*int*) – Integer number of times to iterate over the training data arrays. If unspecified, it will default to 1000.
- **patience** (*int*) – Number of epochs with no improvement. If unspecified, it will default to 20.
- **lr_schedule_min_lr** (*float*) – Lower bound on the learning rate. If unspecified, it will default to 1e-5.
- **lr_schedule_factor** (*float*) – Factor by which the learning rate will be reduced. new_lr = lr * factor. If unspecified, it will default to 0.2.

- **lr_schedule_patience** (*int*) – Number of epochs with no improvement after which learning rate will be reduced. If unspecified, it will default to 5.
- **initial_epoch** (*int*) – Epoch at which to start training (useful for resuming a previous training run). If unspecified, it will default to 0.
- **monitor_partition** (*str*) – Monitor partition.
- **monitor_metric** (*str*) – Monitor metric.
- **log_dir** (*str, optional*) – Logging directory.
- **callbacks** (*list, optional*) – List of callbacks to be called during training.
- **early_stopping** (*bool*) – Whether to activate early stopping.
- **reduce_lr_plateau** (*bool*) – Whether to reduce learning rate on plateau.
- **kwargs** – Arbitrary keyword arguments.

3.1.4 ncem.estimators.EstimatorCVAE

class ncem.estimators.**EstimatorCVAE**(*use_type_cond=True*, *log_transform=False*)

Estimator class for conditional variational autoencoder models. Subclass of EstimatorNoGraph.

Attributes

img_keys_all	Return all image keys.
nodes_idx_all	Return all node indices.
nodes_idx_eval	alias of Dict[str, list]
nodes_idx_test	alias of Dict[str, list]
nodes_idx_train	alias of Dict[str, list]
patient_ids_bytarget	Return patient identifiers by target.
patient_ids_unique	Return unique patient identifiers.

ncem.estimators.EstimatorCVAE.img_keys_all

property EstimatorCVAE.**img_keys_all**

Return all image keys.

Return type
img_keys_all

ncem.estimators.EstimatorCVAE.nodes_idx_all

property EstimatorCVAE.**nodes_idx_all**

Return all node indices.

Return type
nodes_idx_all

ncem.estimators.EstimatorCVAE.nodes_idx_eval

EstimatorCVAE.**nodes_idx_eval**
alias of Dict[str, list]

alias of Dict[str, list] .. autoattribute:: EstimatorCVAE.nodes_idx_eval

ncem.estimators.EstimatorCVAE.nodes_idx_test

EstimatorCVAE.**nodes_idx_test**
alias of Dict[str, list]

alias of Dict[str, list] .. autoattribute:: EstimatorCVAE.nodes_idx_test

ncem.estimators.EstimatorCVAE.nodes_idx_train

EstimatorCVAE.**nodes_idx_train**
alias of Dict[str, list]

alias of Dict[str, list] .. autoattribute:: EstimatorCVAE.nodes_idx_train

ncem.estimators.EstimatorCVAE.patient_ids_bytarget

property EstimatorCVAE.**patient_ids_bytarget: ndarray**
Return patient identifiers by target.

Return type
patient_ids_bytarget

ncem.estimators.EstimatorCVAE.patient_ids_unique

property EstimatorCVAE.**patient_ids_unique: ndarray**
Return unique patient identifiers.

Return type
patient_ids_unique

Methods

evaluate_any(img_keys, node_idx[, batch_size])	Evaluate model on any given data set.
evaluate_any_posterior_sampling(img_keys, ...)	Evaluate model based on resampled dataset for posterior resampling.
evaluate_per_node_type([batch_size])	Evaluate model for each node type seperately.
get_data(data_origin, data_path, radius[, ...])	Get data used in estimator classes.
init_model([optimizer, learning_rate, ...])	Initialize a ModelCVAE object.
predict([batch_size])	Return observed labels and full predictions (including scale model) grouped exactly as in nodes_idx_test.
pretrain_decoder([decoder_epochs, patience, ...])	Pre-train decoder model.
split_data_given(img_keys_test, ...)	Split data by given partition.
split_data_node(test_split, validation_split)	Split nodes randomly into partitions.
split_data_target_cell(target_cell, ...[, seed])	Split nodes randomly into partitions.
train([epochs, epochs_warmup, ...])	Train model.
train_aggressive([aggressive_enc_patience, ...])	Train model aggressive.
train_normal([epochs, patience, ...])	Train model normal.

ncem.estimators.EstimatorCVAE.evaluate_any

EstimatorCVAE.**evaluate_any**(*img_keys*, *node_idx*, *batch_size=1*)

Evaluate model on any given data set.

Parameters

- **img_keys** – Image keys.
- **node_idx** – Nodes indices.
- **batch_size** (*int*) – Number of samples. If unspecified, it will default to 1.

Return type
eval_dict

ncem.estimators.EstimatorCVAE.evaluate_any_posterior_sampling

EstimatorCVAE.**evaluate_any_posterior_sampling**(*img_keys*, *node_idx*, *batch_size=1*)

Evaluate model based on resampled dataset for posterior resampling.

node_1 + domain_1 -> encoder -> z_1 + domain_2 -> decoder -> reconstruction_2.

Parameters

- **img_keys** – Image keys in partition.
- **node_idx** – Dictionary of nodes per image in partition.
- **batch_size** (*int*) – Batch size.

Return type
Tuple of dictionary of evaluated metrics and latent space arrays (z, z_mean, z_log_var).

ncem.estimators.EstimatorCVAE.evaluate_per_node_type

EstimatorCVAE.**evaluate_per_node_type**(*batch_size=1*)

Evaluate model for each node type seperately.

Parameters

batch_size (*int*) – Number of samples. If unspecified, it will default to 1.

Return type

split_per_node_type, evaluation_per_node_type

ncem.estimators.EstimatorCVAE.get_data

EstimatorCVAE.**get_data**(*data_origin*, *data_path*, *radius*, *n_rings=1*, *graph_covar_selection=None*, *node_label_space_id='type'*, *node_feature_space_id='standard'*, *use_covar_node_position=False*, *use_covar_node_label=False*, *use_covar_graph_covar=False*, *domain_type='image'*, *robustness=None*, *robustness_seed=1*, *n_top_genes=None*, *segmentation_robustness=None*, *resimulate_nodes=False*, *resimulate_nodes_w_depdency=False*, *resimulate_nodes_sparsity_rate=0.5*)

Get data used in estimator classes.

Parameters

- **data_origin** (*str*) – Data origin.
- **data_path** (*str*) – Data path.
- **radius** (*int, optional*) – Radius.
- **n_rings** (*int*) – Number of rings of neighbors for grid data.
- **graph_covar_selection** (*list, tuple, optional*) – Selected graph covariates.
- **node_label_space_id** (*str*) – Node label space id.
- **node_feature_space_id** (*str*) – Node feature space id.
- **use_covar_node_position** (*bool*) – Whether to use node position as covariate.
- **use_covar_node_label** (*bool*) – Whether to use node label as covariate.
- **use_covar_graph_covar** (*bool*) – Whether to use graph covariates.
- **domain_type** (*str*) – Covariate that is used as domain.
- **robustness** (*float, optional*) – Optional fraction of images for robustness test.
- **robustness_seed** (*int*) – Seed for robustness analysis
- **n_top_genes** (*int, optional*) – N top genes for highly variable gene selection.
- **segmentation_robustness** (*list, optional*) – Parameters for segmentation robustness fit, float for fraction of nodes and float for signal overflow.
- **resimulate_nodes** (*bool*) –
- **resimulate_nodes_w_depdency** (*bool*) –
- **resimulate_nodes_sparsity_rate** (*float*) –

Raises

ValueError – If sub-selected covar_selection could not be found, *node_label_space_id* or *node_feature_space_id* not recognized

ncem.estimators.EstimatorCVAE.init_model

EstimatorCVAE.**init_model**(*optimizer='adam'*, *learning_rate=0.0001*, *latent_dim=10*, *intermediate_dim_enc=128*, *intermediate_dim_dec=128*, *depth_enc=1*, *depth_dec=1*, *dropout_rate=0.1*, *l2_coef=0.0*, *l1_coef=0.0*, *n_eval_nodes_per_graph=10*, *use_domain=False*, *use_batch_norm=False*, *scale_node_size=True*, *transform_input=False*, *beta=0.01*, *max_beta=1.0*, *pre_warm_up=0*, *output_layer='gaussian'*, ***kwargs*)

Initialize a ModelCVAE object.

Parameters

- **optimizer** (*str*) – Optimizer.
- **learning_rate** (*float*) – Learning rate.
- **latent_dim** (*int*) – Latent dimension.
- **dropout_rate** (*float*) – Dropout rate.
- **l2_coef** (*float*) – l2 regularization coefficient.
- **l1_coef** (*float*) – l1 regularization coefficient.
- **intermediate_dim_enc** (*int*) – Encoder intermediate dimension.
- **depth_enc** (*int*) – Encoder depth.
- **intermediate_dim_dec** (*int*) – Decoder intermediate dimension.
- **depth_dec** (*int*) – Decoder depth.
- **n_eval_nodes_per_graph** (*int*) – Number of nodes per graph.
- **use_domain** (*bool*) – Whether to use domain information.
- **use_batch_norm** (*bool*) – Whether to use batch normalization.
- **scale_node_size** (*bool*) – Whether to scale output layer by node sizes.
- **transform_input** (*bool*) – Whether to transform input.
- **beta** (*float*) – Beta used in BetaScheduler.
- **max_beta** (*float*) – Maximal beta used in BetaScheduler.
- **pre_warm_up** (*int*) – Number of epochs in pre warm up.
- **output_layer** (*str*) – Output layer.
- **kwargs** – Arbitrary keyword arguments.

ncem.estimators.EstimatorCVAE.predict

EstimatorCVAE.**predict**(*batch_size=1*)

Return observed labels and full predictions (including scale model) grouped exactly as in nodes_idx_test.

Parameters

batch_size (*int*) – Number of samples. If unspecified, it will default to 1.

Return type

predict

ncem.estimators.EstimatorCVAE.pretrain_decoder

EstimatorCVAE.**pretrain_decoder**(*decoder_epochs=1000*, *patience=20*, *lr_schedule_min_lr=1e-05*, *lr_schedule_factor=0.2*, *lr_schedule_patience=5*, *initial_epoch=0*, *monitor_partition='val'*, *monitor_metric='loss'*, *log_dir=None*, *callbacks=None*, *early_stopping=True*, *reduce_lr_plateau=True*, ***kwargs*)

Pre-train decoder model.

Use validation loss and maximum number of epochs as termination criteria.

Parameters

- **patience** (*int*) – Number of epochs with no improvement. If unspecified, it will default to 20.
- **lr_schedule_min_lr** (*float*) – Lower bound on the learning rate. If unspecified, it will default to 1e-5.
- **lr_schedule_factor** (*float*) – Factor by which the learning rate will be reduced. new_lr = lr * factor. If unspecified, it will default to 0.2.
- **lr_schedule_patience** (*int*) – Number of epochs with no improvement after which learning rate will be reduced. If unspecified, it will default to 5.
- **initial_epoch** (*int*) – Epoch at which to start training (useful for resuming a previous training run). If unspecified, it will default to 0.
- **monitor_partition** (*str*) – Monitor partition.
- **monitor_metric** (*str*) – Monitor metric.
- **log_dir** (*str, optional*) – Logging directory.
- **callbacks** (*list, optional*) – List of callbacks to be called during training.
- **early_stopping** (*bool*) – Whether to activate early stopping.
- **reduce_lr_plateau** (*bool*) – Whether to reduce learning rate on plateau.
- **decoder_epochs** (*int*) – Integer number of times to iterate over the training data arrays in decoder pretraining. If unspecified, it will default to 1000.
- **kwargs** – Arbitrary keyword arguments.

ncem.estimators.EstimatorCVAE.split_data_given

EstimatorCVAE.**split_data_given**(*img_keys_test*, *img_keys_train*, *img_keys_eval*, *nodes_idx_test*, *nodes_idx_train*, *nodes_idx_eval*)

Split data by given partition.

Parameters

- **img_keys_test** – Test image keys.
- **img_keys_train** – Train image keys.
- **img_keys_eval** – Evaluation image keys.
- **nodes_idx_test** – Test node indices.
- **nodes_idx_train** – Train node indices.
- **nodes_idx_eval** – Evaluation node indices.

ncem.estimators.EstimatorCVAE.split_data_node

EstimatorCVAE.**split_data_node**(*test_split*, *validation_split*, *seed=1*)

Split nodes randomly into partitions.

Parameters

- **test_split** (*float*) – Fraction of total nodes to be in test set.
- **validation_split** (*float*) – Fraction of train-eval nodes to be in validation split.
- **seed** (*int*) – Seed for random selection of observations.

Raises

ValueError – If evaluation or test dataset are empty.

ncem.estimators.EstimatorCVAE.split_data_target_cell

EstimatorCVAE.**split_data_target_cell**(*target_cell*, *test_split*, *validation_split*, *seed=1*)

Split nodes randomly into partitions.

Parameters

- **target_cell** (*str*) – Target cell type.
- **test_split** (*float*) – Fraction of total nodes to be in test set.
- **validation_split** (*float*) – Fraction of train-eval nodes to be in validation split.
- **seed** (*int*) – Seed for random selection of observations.

Raises

ValueError – If evaluation or test dataset are empty.

ncem.estimators.EstimatorCVAE.train

EstimatorCVAE.**train**(*epochs=1000*, *epochs_warmup=0*, *max_steps_per_epoch=20*, *batch_size=16*, *validation_batch_size=16*, *max_validation_steps=10*, *shuffle_buffer_size=10000*, *patience=20*, *lr_schedule_min_lr=1e-05*, *lr_schedule_factor=0.2*, *lr_schedule_patience=5*, *initial_epoch=0*, *monitor_partition='val'*, *monitor_metric='loss'*, *log_dir=None*, *callbacks=None*, *early_stopping=True*, *reduce_lr_plateau=True*, *pretrain_decoder=False*, *decoder_epochs=1000*, *decoder_patience=20*, *decoder_callbacks=None*, *aggressive=False*, *aggressive_enc_patience=10*, *aggressive_epochs=5*, *seed=1234*, ***kwargs*)

Train model.

Use validation loss and maximum number of epochs as termination criteria.

Parameters

- **epochs** (*int*) – Integer number of times to iterate over the training data arrays. If unspecified, it will default to 1000.
- **epochs_warmup** (*int*) – Integer number of times to iterate over the training data arrays in warm up (without early stopping). If unspecified, it will default to 0.
- **max_steps_per_epoch** (*int, optional*) – Maximal steps per epoch. If unspecified, it will default to 20.

- **batch_size** (*int*) – Number of samples per gradient update. If unspecified, it will default to 16.
- **validation_batch_size** (*int*) – Number of samples in validation. If unspecified, it will default to 16.
- **max_validation_steps** (*int*) – Maximal steps per validation. If unspecified, it will default to 10.
- **shuffle_buffer_size** (*int, optional*) – Shuffle buffer size. If unspecified, it will default to 1e4.
- **patience** (*int*) – Number of epochs with no improvement. If unspecified, it will default to 20.
- **lr_schedule_min_lr** (*float*) – Lower bound on the learning rate. If unspecified, it will default to 1e-5.
- **lr_schedule_factor** (*float*) – Factor by which the learning rate will be reduced. new_lr = lr * factor. If unspecified, it will default to 0.2.
- **lr_schedule_patience** (*int*) – Number of epochs with no improvement after which learning rate will be reduced. If unspecified, it will default to 5.
- **initial_epoch** (*int*) – Epoch at which to start training (useful for resuming a previous training run). If unspecified, it will default to 0.
- **monitor_partition** (*str*) – Monitor partition.
- **monitor_metric** (*str*) – Monitor metric.
- **log_dir** (*str, optional*) – Logging directory.
- **callbacks** (*list, optional*) – List of callbacks to be called during training.
- **early_stopping** (*bool*) – Whether to activate early stopping.
- **reduce_lr_plateau** (*bool*) – Whether to reduce learning rate on plateau.
- **pretrain_decoder** (*bool*) – Whether to pretrain the decoder model.
- **decoder_epochs** (*int*) – Integer number of times to iterate over the training data arrays in decoder pretraining. If unspecified, it will default to 1000.
- **decoder_patience** (*int*) – Number of epochs with no improvement in decoder pretraining. If unspecified, it will default to 20.
- **decoder_callbacks** (*list, optional*) – List of callbacks to be called during decoder pretraining.
- **aggressive** (*bool*) – Whether to train aggressive.
- **aggressive_enc_patience** (*int*) – Number of epochs with no improvement in aggressive training. If unspecified, it will default to 10.
- **aggressive_epochs** (*int*) – Integer number of times to iterate over the training data arrays in aggressive training. If unspecified, it will default to 5.
- **seed** (*int*) – Random seed for reproduability.
- **kwargs** – Arbitrary keyword arguments.

ncem.estimators.EstimatorCVAE.train_aggressive

EstimatorCVAE.**train_aggressive**(*aggressive_enc_patience=10*, *aggressive_epochs=5*)

Train model aggressive.

Parameters

- **aggressive_enc_patience** (*int*) – Number of epochs with no improvement in aggressive training. If unspecified, it will default to 10.
- **aggressive_epochs** (*int*) – Integer number of times to iterate over the training data arrays in aggressive training. If unspecified, it will default to 5.

ncem.estimators.EstimatorCVAE.train_normal

EstimatorCVAE.**train_normal**(*epochs=1000*, *patience=20*, *lr_schedule_min_lr=1e-05*, *lr_schedule_factor=0.2*, *lr_schedule_patience=5*, *initial_epoch=0*, *monitor_partition='val'*, *monitor_metric='loss'*, *log_dir=None*, *callbacks=None*, *early_stopping=True*, *reduce_lr_plateau=True*, ***kwargs*)

Train model normal.

Use validation loss and maximum number of epochs as termination criteria.

Parameters

- **epochs** (*int*) – Integer number of times to iterate over the training data arrays. If unspecified, it will default to 1000.
- **patience** (*int*) – Number of epochs with no improvement. If unspecified, it will default to 20.
- **lr_schedule_min_lr** (*float*) – Lower bound on the learning rate. If unspecified, it will default to 1e-5.
- **lr_schedule_factor** (*float*) – Factor by which the learning rate will be reduced. new_lr = lr * factor. If unspecified, it will default to 0.2.
- **lr_schedule_patience** (*int*) – Number of epochs with no improvement after which learning rate will be reduced. If unspecified, it will default to 5.
- **initial_epoch** (*int*) – Epoch at which to start training (useful for resuming a previous training run). If unspecified, it will default to 0.
- **monitor_partition** (*str*) – Monitor partition.
- **monitor_metric** (*str*) – Monitor metric.
- **log_dir** (*str, optional*) – Logging directory.
- **callbacks** (*list, optional*) – List of callbacks to be called during training.
- **early_stopping** (*bool*) – Whether to activate early stopping.
- **reduce_lr_plateau** (*bool*) – Whether to reduce learning rate on plateau.
- **kwargs** – Arbitrary keyword arguments.

Parameters

- **use_type_cond** (*bool*) –
- **log_transform** (*bool*) –

3.1.5 ncem.estimators.EstimatorCVAEncem

class ncem.estimators.**EstimatorCVAEncem**(*cond_type='gcn'*, *use_type_cond=True*, *log_transform=False*)

Estimator class for conditional variational autoencoder NCEM models. Subclass of EstimatorGraph.

Attributes

img_keys_all	Return all image keys.
nodes_idx_all	Return all node indices.
nodes_idx_eval	alias of Dict[str, list]
nodes_idx_test	alias of Dict[str, list]
nodes_idx_train	alias of Dict[str, list]
patient_ids_bytarget	Return patient identifiers by target.
patient_ids_unique	Return unique patient identifiers.

ncem.estimators.EstimatorCVAEncem.img_keys_all

property EstimatorCVAEncem.**img_keys_all**

Return all image keys.

Return type
img_keys_all

ncem.estimators.EstimatorCVAEncem.nodes_idx_all

property EstimatorCVAEncem.**nodes_idx_all**

Return all node indices.

Return type
nodes_idx_all

ncem.estimators.EstimatorCVAEncem.nodes_idx_eval

EstimatorCVAEncem.**nodes_idx_eval**

alias of Dict[str, list]

alias of Dict[str, list] .. autoattribute:: EstimatorCVAEncem.nodes_idx_eval

ncem.estimators.EstimatorCVAEncem.nodes_idx_test

EstimatorCVAEncem.**nodes_idx_test**

alias of Dict[str, list]

alias of Dict[str, list] .. autoattribute:: EstimatorCVAEncem.nodes_idx_test

ncem.estimators.EstimatorCVAEncem.nodes_idx_train

EstimatorCVAEncem.**nodes_idx_train**

alias of Dict[str, list]

alias of Dict[str, list] .. autoattribute:: EstimatorCVAEncem.nodes_idx_train

ncem.estimators.EstimatorCVAEncem.patient_ids_bytarget

property EstimatorCVAEncem.**patient_ids_bytarget: ndarray**

Return patient identifiers by target.

Return type
patient_ids_bytarget

ncem.estimators.EstimatorCVAEncem.patient_ids_unique

property EstimatorCVAEncem.**patient_ids_unique: ndarray**

Return unique patient identifiers.

Return type
patient_ids_unique

Methods

evaluate_any(img_keys, node_idx[, batch_size])	Evaluate model on any given data set.
evaluate_any_posterior_sampling(img_keys, ...)	Evaluate model based on resampled dataset for posterior resampling.
evaluate_per_node_type([batch_size])	Evaluate model for each node type seperately.
get_data(data_origin, data_path, radius[, ...])	Get data used in estimator classes.
init_model([optimizer, learning_rate, ...])	Initialize a ModelCVAEncem object.
predict([batch_size])	Return observed labels and full predictions (including scale model) grouped exactly as in nodes_idx_test.
pretrain_decoder([decoder_epochs, patience, ...])	Pre-train decoder model.
split_data_given(img_keys_test, ...)	Split data by given partition.
split_data_node(test_split, validation_split)	Split nodes randomly into partitions.
split_data_target_cell(target_cell, ...[, seed])	Split nodes randomly into partitions.
train([epochs, epochs_warmup, ...])	Train model.
train_aggressive([aggressive_enc_patience, ...])	Train model aggressive.
train_normal([epochs, patience, ...])	Train model normal.

ncem.estimators.EstimatorCVAEncem.evaluate_any

EstimatorCVAEncem.**evaluate_any**(*img_keys*, *node_idx*, *batch_size=1*)

Evaluate model on any given data set.

Parameters

- **img_keys** – Image keys.
- **node_idx** – Nodes indices.
- **batch_size** (*int*) – Number of samples. If unspecified, it will default to 1.

Return type
eval_dict

ncem.estimators.EstimatorCVAEncem.evaluate_any_posterior_sampling

EstimatorCVAEncem.**evaluate_any_posterior_sampling**(*img_keys*, *node_idx*, *batch_size=1*)

Evaluate model based on resampled dataset for posterior resampling.

node_1 + domain_1 -> encoder -> z_1 + domain_2 -> decoder -> reconstruction_2.

Parameters

- **img_keys** – Image keys in partition.
- **node_idx** – Dictionary of nodes per image in partition.
- **batch_size** (*int*) – Batch size.

Return type
Tuple of dictionary of evaluated metrics and latent space arrays (z, z_mean, z_log_var).

ncem.estimators.EstimatorCVAEncem.evaluate_per_node_type

EstimatorCVAEncem.**evaluate_per_node_type**(*batch_size=1*)

Evaluate model for each node type seperately.

Parameters
batch_size (*int*) – Number of samples. If unspecified, it will default to 1.

Return type
split_per_node_type, evaluation_per_node_type

ncem.estimators.EstimatorCVAEncem.get_data

EstimatorCVAEncem.**get_data**(*data_origin*, *data_path*, *radius*, *n_rings=1*, *graph_covar_selection=None*, *node_label_space_id='type'*, *node_feature_space_id='standard'*, *use_covar_node_position=False*, *use_covar_node_label=False*, *use_covar_graph_covar=False*, *domain_type='image'*, *robustness=None*, *robustness_seed=1*, *n_top_genes=None*, *segmentation_robustness=None*, *resimulate_nodes=False*, *resimulate_nodes_w_depdency=False*, *resimulate_nodes_sparsity_rate=0.5*)

Get data used in estimator classes.

Parameters

- **data_origin** (*str*) – Data origin.
- **data_path** (*str*) – Data path.
- **radius** (*int, optional*) – Radius.
- **n_rings** (*int*) – Number of rings of neighbors for grid data.
- **graph_covar_selection** (*list, tuple, optional*) – Selected graph covariates.
- **node_label_space_id** (*str*) – Node label space id.
- **node_feature_space_id** (*str*) – Node feature space id.
- **use_covar_node_position** (*bool*) – Whether to use node position as covariate.
- **use_covar_node_label** (*bool*) – Whether to use node label as covariate.
- **use_covar_graph_covar** (*bool*) – Whether to use graph covariates.
- **domain_type** (*str*) – Covariate that is used as domain.
- **robustness** (*float, optional*) – Optional fraction of images for robustness test.
- **robustness_seed** (*int*) – Seed for robustness analysis
- **n_top_genes** (*int, optional*) – N top genes for highly variable gene selection.
- **segmentation_robustness** (*list, optional*) – Parameters for segmentation robustness fit, float for fraction of nodes and float for signal overflow.
- **resimulate_nodes** (*bool*) –
- **resimulate_nodes_w_depdency** (*bool*) –
- **resimulate_nodes_sparsity_rate** (*float*) –

Raises

ValueError – If sub-selected covar_selection could not be found, *node_label_space_id* or *node_feature_space_id* not recognized

ncem.estimators.EstimatorCVAEncem.init_model

EstimatorCVAEncem.**init_model**(*optimizer='adam'*, *learning_rate=0.0001*, *latent_dim=8*, *intermediate_dim_enc=128*, *intermediate_dim_dec=128*, *depth_enc=1*, *depth_dec=1*, *dropout_rate=0.1*, *l2_coef=0.0*, *l1_coef=0.0*, *cond_depth=1*, *cond_dim=8*, *cond_dropout_rate=0.1*, *cond_activation='relu'*, *cond_l2_reg=0.0*, *cond_use_bias=False*, *n_eval_nodes_per_graph=32*, *use_domain=False*, *use_batch_norm=False*, *scale_node_size=True*, *transform_input=False*, *beta=0.01*, *max_beta=1.0*, *pre_warm_up=0*, *output_layer='gaussian'*, ***kwargs*)

Initialize a ModelCVAEncem object.

Parameters

- **optimizer** (*str*) – Optimizer.
- **learning_rate** (*float*) – Learning rate.
- **latent_dim** (*int*) – Latent dimension.
- **dropout_rate** (*float*) – Dropout rate.
- **l2_coef** (*float*) – l2 regularization coefficient.

- **l1_coef** (*float*) – l1 regularization coefficient.
- **intermediate_dim_enc** (*int*) – Encoder intermediate dimension.
- **depth_enc** (*int*) – Encoder depth.
- **intermediate_dim_dec** (*int*) – Decoder intermediate dimension.
- **depth_dec** (*int*) – Decoder depth.
- **cond_depth** (*int*) – Graph conditional depth.
- **cond_dim** (*int*) – Graph conditional dimension.
- **cond_dropout_rate** (*float*) – Graph conditional dropout rate.
- **cond_activation** (*str*) – Graph conditional activation.
- **cond_l2_reg** (*float*) – Graph conditional l2 regularization coefficient.
- **cond_use_bias** (*bool*) – Graph conditional use bias.
- **n_eval_nodes_per_graph** (*int*) – Number of nodes per graph.
- **use_domain** (*bool*) – Whether to use domain information.
- **use_batch_norm** (*bool*) – Whether to use batch normalization.
- **scale_node_size** (*bool*) – Whether to scale output layer by node sizes.
- **transform_input** (*bool*) – Whether to transform input.
- **beta** (*float*) – Beta used in BetaScheduler.
- **max_beta** (*float*) – Maximal beta used in BetaScheduler.
- **pre_warm_up** (*int*) – Number of epochs in pre warm up.
- **output_layer** (*str*) – Output layer.
- **kwargs** – Arbitrary keyword arguments.

ncem.estimators.EstimatorCVAEncem.predict

EstimatorCVAEncem.**predict**(*batch_size=1*)

Return observed labels and full predictions (including scale model) grouped exactly as in nodes_idx_test.

Parameters
batch_size (*int*) – Number of samples. If unspecified, it will default to 1.

Return type
predict

ncem.estimators.EstimatorCVAEncem.pretrain_decoder

EstimatorCVAEncem.**pretrain_decoder**(*decoder_epochs=1000*, *patience=20*, *lr_schedule_min_lr=1e-05*, *lr_schedule_factor=0.2*, *lr_schedule_patience=5*, *initial_epoch=0*, *monitor_partition='val'*, *monitor_metric='loss'*, *log_dir=None*, *callbacks=None*, *early_stopping=True*, *reduce_lr_plateau=True*, ***kwargs*)

Pre-train decoder model.

Use validation loss and maximum number of epochs as termination criteria.

Parameters

- **patience** (*int*) – Number of epochs with no improvement. If unspecified, it will default to 20.
- **lr_schedule_min_lr** (*float*) – Lower bound on the learning rate. If unspecified, it will default to 1e-5.
- **lr_schedule_factor** (*float*) – Factor by which the learning rate will be reduced. new_lr = lr * factor. If unspecified, it will default to 0.2.
- **lr_schedule_patience** (*int*) – Number of epochs with no improvement after which learning rate will be reduced. If unspecified, it will default to 5.
- **initial_epoch** (*int*) – Epoch at which to start training (useful for resuming a previous training run). If unspecified, it will default to 0.
- **monitor_partition** (*str*) – Monitor partition.
- **monitor_metric** (*str*) – Monitor metric.
- **log_dir** (*str, optional*) – Logging directory.
- **callbacks** (*list, optional*) – List of callbacks to be called during training.
- **early_stopping** (*bool*) – Whether to activate early stopping.
- **reduce_lr_plateau** (*bool*) – Whether to reduce learning rate on plateau.
- **decoder_epochs** (*int*) – Integer number of times to iterate over the training data arrays in decoder pretraining. If unspecified, it will default to 1000.
- **kwargs** – Arbitrary keyword arguments.

ncem.estimators.EstimatorCVAEncem.split_data_given

EstimatorCVAEncem.**split_data_given**(*img_keys_test*, *img_keys_train*, *img_keys_eval*, *nodes_idx_test*, *nodes_idx_train*, *nodes_idx_eval*)

Split data by given partition.

Parameters

- **img_keys_test** – Test image keys.
- **img_keys_train** – Train image keys.
- **img_keys_eval** – Evaluation image keys.
- **nodes_idx_test** – Test node indices.
- **nodes_idx_train** – Train node indices.
- **nodes_idx_eval** – Evaluation node indices.

ncem.estimators.EstimatorCVAEncem.split_data_node

EstimatorCVAEncem.**split_data_node**(*test_split*, *validation_split*, *seed=1*)

Split nodes randomly into partitions.

Parameters

- **test_split** (*float*) – Fraction of total nodes to be in test set.
- **validation_split** (*float*) – Fraction of train-eval nodes to be in validation split.
- **seed** (*int*) – Seed for random selection of observations.

Raises

ValueError – If evaluation or test dataset are empty.

ncem.estimators.EstimatorCVAEncem.split_data_target_cell

EstimatorCVAEncem.**split_data_target_cell**(*target_cell*, *test_split*, *validation_split*, *seed=1*)

Split nodes randomly into partitions.

Parameters

- **target_cell** (*str*) – Target cell type.
- **test_split** (*float*) – Fraction of total nodes to be in test set.
- **validation_split** (*float*) – Fraction of train-eval nodes to be in validation split.
- **seed** (*int*) – Seed for random selection of observations.

Raises

ValueError – If evaluation or test dataset are empty.

ncem.estimators.EstimatorCVAEncem.train

EstimatorCVAEncem.**train**(*epochs=1000*, *epochs_warmup=0*, *max_steps_per_epoch=20*, *batch_size=16*, *validation_batch_size=16*, *max_validation_steps=10*, *shuffle_buffer_size=10000*, *patience=20*, *lr_schedule_min_lr=1e-05*, *lr_schedule_factor=0.2*, *lr_schedule_patience=5*, *initial_epoch=0*, *monitor_partition='val'*, *monitor_metric='loss'*, *log_dir=None*, *callbacks=None*, *early_stopping=True*, *reduce_lr_plateau=True*, *pretrain_decoder=False*, *decoder_epochs=1000*, *decoder_patience=20*, *decoder_callbacks=None*, *aggressive=False*, *aggressive_enc_patience=10*, *aggressive_epochs=5*, *seed=1234*, ***kwargs*)

Train model.

Use validation loss and maximum number of epochs as termination criteria.

Parameters

- **epochs** (*int*) – Integer number of times to iterate over the training data arrays. If unspecified, it will default to 1000.
- **epochs_warmup** (*int*) – Integer number of times to iterate over the training data arrays in warm up (without early stopping). If unspecified, it will default to 0.
- **max_steps_per_epoch** (*int, optional*) – Maximal steps per epoch. If unspecified, it will default to 20.

- **batch_size** (*int*) – Number of samples per gradient update. If unspecified, it will default to 16.
- **validation_batch_size** (*int*) – Number of samples in validation. If unspecified, it will default to 16.
- **max_validation_steps** (*int*) – Maximal steps per validation. If unspecified, it will default to 10.
- **shuffle_buffer_size** (*int, optional*) – Shuffle buffer size. If unspecified, it will default to 1e4.
- **patience** (*int*) – Number of epochs with no improvement. If unspecified, it will default to 20.
- **lr_schedule_min_lr** (*float*) – Lower bound on the learning rate. If unspecified, it will default to 1e-5.
- **lr_schedule_factor** (*float*) – Factor by which the learning rate will be reduced. new_lr = lr * factor. If unspecified, it will default to 0.2.
- **lr_schedule_patience** (*int*) – Number of epochs with no improvement after which learning rate will be reduced. If unspecified, it will default to 5.
- **initial_epoch** (*int*) – Epoch at which to start training (useful for resuming a previous training run). If unspecified, it will default to 0.
- **monitor_partition** (*str*) – Monitor partition.
- **monitor_metric** (*str*) – Monitor metric.
- **log_dir** (*str, optional*) – Logging directory.
- **callbacks** (*list, optional*) – List of callbacks to be called during training.
- **early_stopping** (*bool*) – Whether to activate early stopping.
- **reduce_lr_plateau** (*bool*) – Whether to reduce learning rate on plateau.
- **pretrain_decoder** (*bool*) – Whether to pretrain the decoder model.
- **decoder_epochs** (*int*) – Integer number of times to iterate over the training data arrays in decoder pretraining. If unspecified, it will default to 1000.
- **decoder_patience** (*int*) – Number of epochs with no improvement in decoder pretraining. If unspecified, it will default to 20.
- **decoder_callbacks** (*list, optional*) – List of callbacks to be called during decoder pretraining.
- **aggressive** (*bool*) – Whether to train aggressive.
- **aggressive_enc_patience** (*int*) – Number of epochs with no improvement in aggressive training. If unspecified, it will default to 10.
- **aggressive_epochs** (*int*) – Integer number of times to iterate over the training data arrays in aggressive training. If unspecified, it will default to 5.
- **seed** (*int*) – Random seed for reproduability.
- **kwargs** – Arbitrary keyword arguments.

ncem.estimators.EstimatorCVAEncem.train_aggressive

EstimatorCVAEncem.**train_aggressive**(*aggressive_enc_patience=10*, *aggressive_epochs=5*)

Train model aggressive.

Parameters

- **aggressive_enc_patience** (*int*) – Number of epochs with no improvement in aggressive training. If unspecified, it will default to 10.
- **aggressive_epochs** (*int*) – Integer number of times to iterate over the training data arrays in aggressive training. If unspecified, it will default to 5.

ncem.estimators.EstimatorCVAEncem.train_normal

EstimatorCVAEncem.**train_normal**(*epochs=1000*, *patience=20*, *lr_schedule_min_lr=1e-05*, *lr_schedule_factor=0.2*, *lr_schedule_patience=5*, *initial_epoch=0*, *monitor_partition='val'*, *monitor_metric='loss'*, *log_dir=None*, *callbacks=None*, *early_stopping=True*, *reduce_lr_plateau=True*, ***kwargs*)

Train model normal.

Use validation loss and maximum number of epochs as termination criteria.

Parameters

- **epochs** (*int*) – Integer number of times to iterate over the training data arrays. If unspecified, it will default to 1000.
- **patience** (*int*) – Number of epochs with no improvement. If unspecified, it will default to 20.
- **lr_schedule_min_lr** (*float*) – Lower bound on the learning rate. If unspecified, it will default to 1e-5.
- **lr_schedule_factor** (*float*) – Factor by which the learning rate will be reduced. new_lr = lr * factor. If unspecified, it will default to 0.2.
- **lr_schedule_patience** (*int*) – Number of epochs with no improvement after which learning rate will be reduced. If unspecified, it will default to 5.
- **initial_epoch** (*int*) – Epoch at which to start training (useful for resuming a previous training run). If unspecified, it will default to 0.
- **monitor_partition** (*str*) – Monitor partition.
- **monitor_metric** (*str*) – Monitor metric.
- **log_dir** (*str, optional*) – Logging directory.
- **callbacks** (*list, optional*) – List of callbacks to be called during training.
- **early_stopping** (*bool*) – Whether to activate early stopping.
- **reduce_lr_plateau** (*bool*) – Whether to reduce learning rate on plateau.
- **kwargs** – Arbitrary keyword arguments.

Parameters

- **cond_type** (*str*) –
- **use_type_cond** (*bool*) –

- **log_transform** (*bool*) –

3.1.6 ncem.estimators.EstimatorED

class ncem.estimators.**EstimatorED**(*use_type_cond=True*, *log_transform=False*)

Estimator class for encoder-decoder models. Subclass of EstimatorNoGraph.

Attributes

img_keys_all	Return all image keys.
nodes_idx_all	Return all node indices.
nodes_idx_eval	alias of Dict[str, list]
nodes_idx_test	alias of Dict[str, list]
nodes_idx_train	alias of Dict[str, list]
patient_ids_bytarget	Return patient identifiers by target.
patient_ids_unique	Return unique patient identifiers.

ncem.estimators.EstimatorED.img_keys_all

property EstimatorED.**img_keys_all**

Return all image keys.

Return type
img_keys_all

ncem.estimators.EstimatorED.nodes_idx_all

property EstimatorED.**nodes_idx_all**

Return all node indices.

Return type
nodes_idx_all

ncem.estimators.EstimatorED.nodes_idx_eval

EstimatorED.**nodes_idx_eval**

alias of Dict[str, list]

alias of Dict[str, list] .. autoattribute:: EstimatorED.nodes_idx_eval

ncem.estimators.EstimatorED.nodes_idx_test

EstimatorED.**nodes_idx_test**

alias of Dict[str, list]

alias of Dict[str, list] .. autoattribute:: EstimatorED.nodes_idx_test

ncem.estimators.EstimatorED.nodes_idx_train

EstimatorED.**nodes_idx_train**

alias of Dict[str, list]

alias of Dict[str, list] .. autoattribute:: EstimatorED.nodes_idx_train

ncem.estimators.EstimatorED.patient_ids_bytarget

property EstimatorED.**patient_ids_bytarget: ndarray**

Return patient identifiers by target.

Return type
patient_ids_bytarget

ncem.estimators.EstimatorED.patient_ids_unique

property EstimatorED.**patient_ids_unique: ndarray**

Return unique patient identifiers.

Return type
patient_ids_unique

Methods

evaluate_any(img_keys, node_idx[, batch_size])	Evaluate model on any given data set.
evaluate_per_node_type([batch_size])	Evaluate model for each node type seperately.
get_data(data_origin, data_path, radius[, ...])	Get data used in estimator classes.
init_model([optimizer, learning_rate, ...])	Initialize a ModelED object.
predict([batch_size])	Return observed labels and full predictions (including scale model) grouped exactly as in nodes_idx_test.
pretrain_decoder([decoder_epochs, patience, ...])	Pre-train decoder model.
split_data_given(img_keys_test, ...)	Split data by given partition.
split_data_node(test_split, validation_split)	Split nodes randomly into partitions.
split_data_target_cell(target_cell, ...[, seed])	Split nodes randomly into partitions.
train([epochs, epochs_warmup, ...])	Train model.
train_aggressive([aggressive_enc_patience, ...])	Train model aggressive.
train_normal([epochs, patience, ...])	Train model normal.

ncem.estimators.EstimatorED.evaluate_any

EstimatorED.**evaluate_any**(*img_keys*, *node_idx*, *batch_size=1*)

Evaluate model on any given data set.

Parameters

- **img_keys** – Image keys.
- **node_idx** – Nodes indices.
- **batch_size** (*int*) – Number of samples. If unspecified, it will default to 1.

Return type
eval_dict

ncem.estimators.EstimatorED.evaluate_per_node_type

EstimatorED.**evaluate_per_node_type**(*batch_size=1*)

Evaluate model for each node type seperately.

Parameters
batch_size (*int*) – Number of samples. If unspecified, it will default to 1.

Return type
split_per_node_type, evaluation_per_node_type

ncem.estimators.EstimatorED.get_data

EstimatorED.**get_data**(*data_origin*, *data_path*, *radius*, *n_rings=1*, *graph_covar_selection=None*, *node_label_space_id='type'*, *node_feature_space_id='standard'*, *use_covar_node_position=False*, *use_covar_node_label=False*, *use_covar_graph_covar=False*, *domain_type='image'*, *robustness=None*, *robustness_seed=1*, *n_top_genes=None*, *segmentation_robustness=None*, *resimulate_nodes=False*, *resimulate_nodes_w_depdency=False*, *resimulate_nodes_sparsity_rate=0.5*)

Get data used in estimator classes.

Parameters

- **data_origin** (*str*) – Data origin.
- **data_path** (*str*) – Data path.
- **radius** (*int, optional*) – Radius.
- **n_rings** (*int*) – Number of rings of neighbors for grid data.
- **graph_covar_selection** (*list, tuple, optional*) – Selected graph covariates.
- **node_label_space_id** (*str*) – Node label space id.
- **node_feature_space_id** (*str*) – Node feature space id.
- **use_covar_node_position** (*bool*) – Whether to use node position as covariate.
- **use_covar_node_label** (*bool*) – Whether to use node label as covariate.
- **use_covar_graph_covar** (*bool*) – Whether to use graph covariates.
- **domain_type** (*str*) – Covariate that is used as domain.

- **robustness** (*float, optional*) – Optional fraction of images for robustness test.
- **robustness_seed** (*int*) – Seed for robustness analysis
- **n_top_genes** (*int, optional*) – N top genes for highly variable gene selection.
- **segmentation_robustness** (*list, optional*) – Parameters for segmentation robustness fit, float for fraction of nodes and float for signal overflow.
- **resimulate_nodes** (*bool*) –
- **resimulate_nodes_w_depdency** (*bool*) –
- **resimulate_nodes_sparsity_rate** (*float*) –

Raises

ValueError – If sub-selected covar_selection could not be found, *node_label_space_id* or *node_feature_space_id* not recognized

ncem.estimators.EstimatorED.init_model

EstimatorED.**init_model**(*optimizer='adam'*, *learning_rate=0.0001*, *latent_dim=10*, *dropout_rate=0.1*, *l2_coef=0.0*, *l1_coef=0.0*, *enc_intermediate_dim=128*, *enc_depth=2*, *dec_intermediate_dim=128*, *dec_depth=2*, *n_eval_nodes_per_graph=32*, *use_domain=False*, *scale_node_size=True*, *beta=0.01*, *max_beta=1.0*, *pre_warm_up=0*, *output_layer='gaussian'*, ***kwargs*)

Initialize a ModelED object.

Parameters

- **optimizer** (*str*) – Optimizer.
- **learning_rate** (*float*) – Learning rate.
- **latent_dim** (*int*) – Latent dimension.
- **dropout_rate** (*float*) – Dropout rate.
- **l2_coef** (*float*) – l2 regularization coefficient.
- **l1_coef** (*float*) – l1 regularization coefficient.
- **enc_intermediate_dim** (*int*) – Encoder intermediate dimension.
- **enc_depth** (*int*) – Encoder depth.
- **dec_intermediate_dim** (*int*) – Decoder intermediate dimension.
- **dec_depth** (*int*) – Decoder depth.
- **n_eval_nodes_per_graph** (*int*) – Number of nodes per graph.
- **use_domain** (*bool*) – Whether to use domain information.
- **scale_node_size** (*bool*) – Whether to scale output layer by node sizes.
- **beta** (*float*) – Beta used in BetaScheduler.
- **max_beta** (*float*) – Maximal beta used in BetaScheduler.
- **pre_warm_up** (*int*) – Number of epochs in pre warm up.
- **output_layer** (*str*) – Output layer.
- **kwargs** – Arbitrary keyword arguments.

ncem.estimators.EstimatorED.predict

EstimatorED.**predict**(*batch_size=1*)

Return observed labels and full predictions (including scale model) grouped exactly as in nodes_idx_test.

Parameters
batch_size (*int*) – Number of samples. If unspecified, it will default to 1.

Return type
predict

ncem.estimators.EstimatorED.pretrain_decoder

EstimatorED.**pretrain_decoder**(*decoder_epochs=1000*, *patience=20*, *lr_schedule_min_lr=1e-05*, *lr_schedule_factor=0.2*, *lr_schedule_patience=5*, *initial_epoch=0*, *monitor_partition='val'*, *monitor_metric='loss'*, *log_dir=None*, *callbacks=None*, *early_stopping=True*, *reduce_lr_plateau=True*, ***kwargs*)

Pre-train decoder model.

Use validation loss and maximum number of epochs as termination criteria.

Parameters

- **patience** (*int*) – Number of epochs with no improvement. If unspecified, it will default to 20.
- **lr_schedule_min_lr** (*float*) – Lower bound on the learning rate. If unspecified, it will default to 1e-5.
- **lr_schedule_factor** (*float*) – Factor by which the learning rate will be reduced. new_lr = lr * factor. If unspecified, it will default to 0.2.
- **lr_schedule_patience** (*int*) – Number of epochs with no improvement after which learning rate will be reduced. If unspecified, it will default to 5.
- **initial_epoch** (*int*) – Epoch at which to start training (useful for resuming a previous training run). If unspecified, it will default to 0.
- **monitor_partition** (*str*) – Monitor partition.
- **monitor_metric** (*str*) – Monitor metric.
- **log_dir** (*str, optional*) – Logging directory.
- **callbacks** (*list, optional*) – List of callbacks to be called during training.
- **early_stopping** (*bool*) – Whether to activate early stopping.
- **reduce_lr_plateau** (*bool*) – Whether to reduce learning rate on plateau.
- **decoder_epochs** (*int*) – Integer number of times to iterate over the training data arrays in decoder pretraining. If unspecified, it will default to 1000.
- **kwargs** – Arbitrary keyword arguments.

ncem.estimators.EstimatorED.split_data_given

EstimatorED.**split_data_given**(*img_keys_test*, *img_keys_train*, *img_keys_eval*, *nodes_idx_test*, *nodes_idx_train*, *nodes_idx_eval*)

Split data by given partition.

Parameters

- **img_keys_test** – Test image keys.
- **img_keys_train** – Train image keys.
- **img_keys_eval** – Evaluation image keys.
- **nodes_idx_test** – Test node indices.
- **nodes_idx_train** – Train node indices.
- **nodes_idx_eval** – Evaluation node indices.

ncem.estimators.EstimatorED.split_data_node

EstimatorED.**split_data_node**(*test_split*, *validation_split*, *seed=1*)

Split nodes randomly into partitions.

Parameters

- **test_split** (*float*) – Fraction of total nodes to be in test set.
- **validation_split** (*float*) – Fraction of train-eval nodes to be in validation split.
- **seed** (*int*) – Seed for random selection of observations.

Raises

ValueError – If evaluation or test dataset are empty.

ncem.estimators.EstimatorED.split_data_target_cell

EstimatorED.**split_data_target_cell**(*target_cell*, *test_split*, *validation_split*, *seed=1*)

Split nodes randomly into partitions.

Parameters

- **target_cell** (*str*) – Target cell type.
- **test_split** (*float*) – Fraction of total nodes to be in test set.
- **validation_split** (*float*) – Fraction of train-eval nodes to be in validation split.
- **seed** (*int*) – Seed for random selection of observations.

Raises

ValueError – If evaluation or test dataset are empty.

ncem.estimators.EstimatorED.train

EstimatorED.**train**(*epochs=1000*, *epochs_warmup=0*, *max_steps_per_epoch=20*, *batch_size=16*, *validation_batch_size=16*, *max_validation_steps=10*, *shuffle_buffer_size=10000*, *patience=20*, *lr_schedule_min_lr=1e-05*, *lr_schedule_factor=0.2*, *lr_schedule_patience=5*, *initial_epoch=0*, *monitor_partition='val'*, *monitor_metric='loss'*, *log_dir=None*, *callbacks=None*, *early_stopping=True*, *reduce_lr_plateau=True*, *pretrain_decoder=False*, *decoder_epochs=1000*, *decoder_patience=20*, *decoder_callbacks=None*, *aggressive=False*, *aggressive_enc_patience=10*, *aggressive_epochs=5*, *seed=1234*, ***kwargs*)

Train model.

Use validation loss and maximum number of epochs as termination criteria.

Parameters

- **epochs** (*int*) – Integer number of times to iterate over the training data arrays. If unspecified, it will default to 1000.
- **epochs_warmup** (*int*) – Integer number of times to iterate over the training data arrays in warm up (without early stopping). If unspecified, it will default to 0.
- **max_steps_per_epoch** (*int, optional*) – Maximal steps per epoch. If unspecified, it will default to 20.
- **batch_size** (*int*) – Number of samples per gradient update. If unspecified, it will default to 16.
- **validation_batch_size** (*int*) – Number of samples in validation. If unspecified, it will default to 16.
- **max_validation_steps** (*int*) – Maximal steps per validation. If unspecified, it will default to 10.
- **shuffle_buffer_size** (*int, optional*) – Shuffle buffer size. If unspecified, it will default to 1e4.
- **patience** (*int*) – Number of epochs with no improvement. If unspecified, it will default to 20.
- **lr_schedule_min_lr** (*float*) – Lower bound on the learning rate. If unspecified, it will default to 1e-5.
- **lr_schedule_factor** (*float*) – Factor by which the learning rate will be reduced. new_lr = lr * factor. If unspecified, it will default to 0.2.
- **lr_schedule_patience** (*int*) – Number of epochs with no improvement after which learning rate will be reduced. If unspecified, it will default to 5.
- **initial_epoch** (*int*) – Epoch at which to start training (useful for resuming a previous training run). If unspecified, it will default to 0.
- **monitor_partition** (*str*) – Monitor partition.
- **monitor_metric** (*str*) – Monitor metric.
- **log_dir** (*str, optional*) – Logging directory.
- **callbacks** (*list, optional*) – List of callbacks to be called during training.
- **early_stopping** (*bool*) – Whether to activate early stopping.
- **reduce_lr_plateau** (*bool*) – Whether to reduce learning rate on plateau.

- **pretrain_decoder** (*bool*) – Whether to pretrain the decoder model.
- **decoder_epochs** (*int*) – Integer number of times to iterate over the training data arrays in decoder pretraining. If unspecified, it will default to 1000.
- **decoder_patience** (*int*) – Number of epochs with no improvement in decoder pretraining. If unspecified, it will default to 20.
- **decoder_callbacks** (*list, optional*) – List of callbacks to be called during decoder pretraining.
- **aggressive** (*bool*) – Whether to train aggressive.
- **aggressive_enc_patience** (*int*) – Number of epochs with no improvement in aggressive training. If unspecified, it will default to 10.
- **aggressive_epochs** (*int*) – Integer number of times to iterate over the training data arrays in aggressive training. If unspecified, it will default to 5.
- **seed** (*int*) – Random seed for reproduability.
- **kwargs** – Arbitrary keyword arguments.

ncem.estimators.EstimatorED.train_aggressive

EstimatorED.**train_aggressive**(*aggressive_enc_patience=10*, *aggressive_epochs=5*)

Train model aggressive.

Parameters

- **aggressive_enc_patience** (*int*) – Number of epochs with no improvement in aggressive training. If unspecified, it will default to 10.
- **aggressive_epochs** (*int*) – Integer number of times to iterate over the training data arrays in aggressive training. If unspecified, it will default to 5.

ncem.estimators.EstimatorED.train_normal

EstimatorED.**train_normal**(*epochs=1000*, *patience=20*, *lr_schedule_min_lr=1e-05*, *lr_schedule_factor=0.2*, *lr_schedule_patience=5*, *initial_epoch=0*, *monitor_partition='val'*, *monitor_metric='loss'*, *log_dir=None*, *callbacks=None*, *early_stopping=True*, *reduce_lr_plateau=True*, ***kwargs*)

Train model normal.

Use validation loss and maximum number of epochs as termination criteria.

Parameters

- **epochs** (*int*) – Integer number of times to iterate over the training data arrays. If unspecified, it will default to 1000.
- **patience** (*int*) – Number of epochs with no improvement. If unspecified, it will default to 20.
- **lr_schedule_min_lr** (*float*) – Lower bound on the learning rate. If unspecified, it will default to 1e-5.
- **lr_schedule_factor** (*float*) – Factor by which the learning rate will be reduced. new_lr = lr * factor. If unspecified, it will default to 0.2.

- **lr_schedule_patience** (*int*) – Number of epochs with no improvement after which learning rate will be reduced. If unspecified, it will default to 5.
- **initial_epoch** (*int*) – Epoch at which to start training (useful for resuming a previous training run). If unspecified, it will default to 0.
- **monitor_partition** (*str*) – Monitor partition.
- **monitor_metric** (*str*) – Monitor metric.
- **log_dir** (*str, optional*) – Logging directory.
- **callbacks** (*list, optional*) – List of callbacks to be called during training.
- **early_stopping** (*bool*) – Whether to activate early stopping.
- **reduce_lr_plateau** (*bool*) – Whether to reduce learning rate on plateau.
- **kwargs** – Arbitrary keyword arguments.

Parameters

- **use_type_cond** (*bool*) –
- **log_transform** (*bool*) –

3.1.7 ncem.estimators.EstimatorEDncem

class ncem.estimators.**EstimatorEDncem**(*cond_type='gcn'*, *use_type_cond=True*, *log_transform=False*)

Estimator class for encoder-decoder NCEM models. Subclass of EstimatorGraph.

Attributes

img_keys_all	Return all image keys.
nodes_idx_all	Return all node indices.
nodes_idx_eval	alias of Dict[str, list]
nodes_idx_test	alias of Dict[str, list]
nodes_idx_train	alias of Dict[str, list]
patient_ids_bytarget	Return patient identifiers by target.
patient_ids_unique	Return unique patient identifiers.

ncem.estimators.EstimatorEDncem.img_keys_all

property EstimatorEDncem.**img_keys_all**

Return all image keys.

Return type
img_keys_all

ncem.estimators.EstimatorEDncem.nodes_idx_all

property EstimatorEDncem.**nodes_idx_all**

Return all node indices.

Return type
nodes_idx_all

ncem.estimators.EstimatorEDncem.nodes_idx_eval

EstimatorEDncem.**nodes_idx_eval**

alias of Dict[str, list]

alias of Dict[str, list] .. autoattribute:: EstimatorEDncem.nodes_idx_eval

ncem.estimators.EstimatorEDncem.nodes_idx_test

EstimatorEDncem.**nodes_idx_test**

alias of Dict[str, list]

alias of Dict[str, list] .. autoattribute:: EstimatorEDncem.nodes_idx_test

ncem.estimators.EstimatorEDncem.nodes_idx_train

EstimatorEDncem.**nodes_idx_train**

alias of Dict[str, list]

alias of Dict[str, list] .. autoattribute:: EstimatorEDncem.nodes_idx_train

ncem.estimators.EstimatorEDncem.patient_ids_bytarget

property EstimatorEDncem.**patient_ids_bytarget: ndarray**

Return patient identifiers by target.

Return type
patient_ids_bytarget

ncem.estimators.EstimatorEDncem.patient_ids_unique

property EstimatorEDncem.**patient_ids_unique: ndarray**

Return unique patient identifiers.

Return type
patient_ids_unique

Methods

evaluate_any(img_keys, node_idx[, batch_size])	Evaluate model on any given data set.
evaluate_per_node_type([batch_size])	Evaluate model for each node type seperately.
get_data(data_origin, data_path, radius[, ...])	Get data used in estimator classes.
get_decoding_weights()	
init_model([optimizer, learning_rate, ...])	Initialize a ModelEDncem object.
predict([batch_size])	Return observed labels and full predictions (including scale model) grouped exactly as in nodes_idx_test.
predict_embedding_any(img_keys, node_idx[, ...])	Predict embedding on any given data set.
pretrain_decoder([decoder_epochs, patience, ...])	Pre-train decoder model.
split_data_given(img_keys_test, ...)	Split data by given partition.
split_data_node(test_split, validation_split)	Split nodes randomly into partitions.
split_data_target_cell(target_cell, ...[, seed])	Split nodes randomly into partitions.
train([epochs, epochs_warmup, ...])	Train model.
train_aggressive([aggressive_enc_patience, ...])	Train model aggressive.
train_normal([epochs, patience, ...])	Train model normal.

ncem.estimators.EstimatorEDncem.evaluate_any

EstimatorEDncem.**evaluate_any**(*img_keys*, *node_idx*, *batch_size=1*)

Evaluate model on any given data set.

Parameters

- **img_keys** – Image keys.
- **node_idx** – Nodes indices.
- **batch_size** (*int*) – Number of samples. If unspecified, it will default to 1.

Return type
eval_dict

ncem.estimators.EstimatorEDncem.evaluate_per_node_type

EstimatorEDncem.**evaluate_per_node_type**(*batch_size=1*)

Evaluate model for each node type seperately.

Parameters
batch_size (*int*) – Number of samples. If unspecified, it will default to 1.

Return type
split_per_node_type, evaluation_per_node_type

ncem.estimators.EstimatorEDncem.get_data

EstimatorEDncem.**get_data**(*data_origin*, *data_path*, *radius*, *n_rings=1*, *graph_covar_selection=None*, *node_label_space_id='type'*, *node_feature_space_id='standard'*, *use_covar_node_position=False*, *use_covar_node_label=False*, *use_covar_graph_covar=False*, *domain_type='image'*, *robustness=None*, *robustness_seed=1*, *n_top_genes=None*, *segmentation_robustness=None*, *resimulate_nodes=False*, *resimulate_nodes_w_depdency=False*, *resimulate_nodes_sparsity_rate=0.5*)

Get data used in estimator classes.

Parameters

- **data_origin** (*str*) – Data origin.
- **data_path** (*str*) – Data path.
- **radius** (*int, optional*) – Radius.
- **n_rings** (*int*) – Number of rings of neighbors for grid data.
- **graph_covar_selection** (*list, tuple, optional*) – Selected graph covariates.
- **node_label_space_id** (*str*) – Node label space id.
- **node_feature_space_id** (*str*) – Node feature space id.
- **use_covar_node_position** (*bool*) – Whether to use node position as covariate.
- **use_covar_node_label** (*bool*) – Whether to use node label as covariate.
- **use_covar_graph_covar** (*bool*) – Whether to use graph covariates.
- **domain_type** (*str*) – Covariate that is used as domain.
- **robustness** (*float, optional*) – Optional fraction of images for robustness test.
- **robustness_seed** (*int*) – Seed for robustness analysis
- **n_top_genes** (*int, optional*) – N top genes for highly variable gene selection.
- **segmentation_robustness** (*list, optional*) – Parameters for segmentation robustness fit, float for fraction of nodes and float for signal overflow.
- **resimulate_nodes** (*bool*) –
- **resimulate_nodes_w_depdency** (*bool*) –
- **resimulate_nodes_sparsity_rate** (*float*) –

Raises

ValueError – If sub-selected covar_selection could not be found, *node_label_space_id* or *node_feature_space_id* not recognized

ncem.estimators.EstimatorEDncem.get_decoding_weights

EstimatorEDncem.**get_decoding_weights**()

ncem.estimators.EstimatorEDncem.init_model

EstimatorEDncem.**init_model**(*optimizer='adam'*, *learning_rate=0.0001*, *latent_dim=10*, *dropout_rate=0.1*, *l2_coef=0.0*, *l1_coef=0.0*, *enc_intermediate_dim=128*, *enc_depth=2*, *dec_intermediate_dim=128*, *dec_depth=2*, *cond_depth=1*, *cond_dim=8*, *cond_dropout_rate=0.1*, *cond_activation='relu'*, *cond_l2_reg=0.0*, *cond_use_bias=False*, *n_eval_nodes_per_graph=32*, *use_domain=False*, *scale_node_size=True*, *beta=0.01*, *max_beta=1.0*, *pre_warm_up=0*, *output_layer='gaussian'*, ***kwargs*)

Initialize a ModelEDncem object.

Parameters

- **optimizer** (*str*) – Optimizer.
- **learning_rate** (*float*) – Learning rate.
- **latent_dim** (*int*) – Latent dimension.
- **dropout_rate** (*float*) – Dropout.
- **l2_coef** (*float*) – l2 regularization coefficient.
- **l1_coef** (*float*) – l1 regularization coefficient.
- **enc_intermediate_dim** (*int*) – Encoder intermediate dimension.
- **enc_depth** (*int*) – Encoder depth.
- **dec_intermediate_dim** (*int*) – Decoder intermediate dimension.
- **dec_depth** (*int*) – Decoder depth.
- **cond_depth** (*int*) – Graph conditional depth.
- **cond_dim** (*int*) – Graph conditional dimension.
- **cond_dropout_rate** (*float*) – Graph conditional dropout rate.
- **cond_activation** (*str*) – Graph conditional activation.
- **cond_l2_reg** (*float*) – Graph conditional l2 regularization coefficient.
- **cond_use_bias** (*bool*) – Graph conditional use bias.
- **n_eval_nodes_per_graph** (*int*) – Number of nodes per graph.
- **use_domain** (*bool*) – Whether to use domain information.
- **scale_node_size** (*bool*) – Whether to scale output layer by node sizes.
- **beta** (*float*) – Beta used in BetaScheduler.
- **max_beta** (*float*) – Maximal beta used in BetaScheduler.
- **pre_warm_up** (*int*) – Number of epochs in pre warm up.
- **output_layer** (*str*) – Output layer.
- **kwargs** – Arbitrary keyword arguments.

ncem.estimators.EstimatorEDncem.predict

EstimatorEDncem.**predict**(*batch_size=1*)

Return observed labels and full predictions (including scale model) grouped exactly as in nodes_idx_test.

Parameters
batch_size (*int*) – Number of samples. If unspecified, it will default to 1.

Return type
predict

ncem.estimators.EstimatorEDncem.predict_embedding_any

EstimatorEDncem.**predict_embedding_any**(*img_keys*, *node_idx*, *batch_size=1*)

Predict embedding on any given data set.

Parameters

- **img_keys** – Image keys.
- **node_idx** – Nodes indices.
- **batch_size** (*int*) – Number of samples. If unspecified, it will default to 1.

Return type
eval_dict

ncem.estimators.EstimatorEDncem.pretrain_decoder

EstimatorEDncem.**pretrain_decoder**(*decoder_epochs=1000*, *patience=20*, *lr_schedule_min_lr=1e-05*, *lr_schedule_factor=0.2*, *lr_schedule_patience=5*, *initial_epoch=0*, *monitor_partition='val'*, *monitor_metric='loss'*, *log_dir=None*, *callbacks=None*, *early_stopping=True*, *reduce_lr_plateau=True*, ***kwargs*)

Pre-train decoder model.

Use validation loss and maximum number of epochs as termination criteria.

Parameters

- **patience** (*int*) – Number of epochs with no improvement. If unspecified, it will default to 20.
- **lr_schedule_min_lr** (*float*) – Lower bound on the learning rate. If unspecified, it will default to 1e-5.
- **lr_schedule_factor** (*float*) – Factor by which the learning rate will be reduced. new_lr = lr * factor. If unspecified, it will default to 0.2.
- **lr_schedule_patience** (*int*) – Number of epochs with no improvement after which learning rate will be reduced. If unspecified, it will default to 5.
- **initial_epoch** (*int*) – Epoch at which to start training (useful for resuming a previous training run). If unspecified, it will default to 0.
- **monitor_partition** (*str*) – Monitor partition.
- **monitor_metric** (*str*) – Monitor metric.
- **log_dir** (*str, optional*) – Logging directory.

- **callbacks** (*list, optional*) – List of callbacks to be called during training.
- **early_stopping** (*bool*) – Whether to activate early stopping.
- **reduce_lr_plateau** (*bool*) – Whether to reduce learning rate on plateau.
- **decoder_epochs** (*int*) – Integer number of times to iterate over the training data arrays in decoder pretraining. If unspecified, it will default to 1000.
- **kwargs** – Arbitrary keyword arguments.

ncem.estimators.EstimatorEDncem.split_data_given

EstimatorEDncem.**split_data_given**(*img_keys_test*, *img_keys_train*, *img_keys_eval*, *nodes_idx_test*, *nodes_idx_train*, *nodes_idx_eval*)

Split data by given partition.

Parameters

- **img_keys_test** – Test image keys.
- **img_keys_train** – Train image keys.
- **img_keys_eval** – Evaluation image keys.
- **nodes_idx_test** – Test node indices.
- **nodes_idx_train** – Train node indices.
- **nodes_idx_eval** – Evaluation node indices.

ncem.estimators.EstimatorEDncem.split_data_node

EstimatorEDncem.**split_data_node**(*test_split*, *validation_split*, *seed=1*)

Split nodes randomly into partitions.

Parameters

- **test_split** (*float*) – Fraction of total nodes to be in test set.
- **validation_split** (*float*) – Fraction of train-eval nodes to be in validation split.
- **seed** (*int*) – Seed for random selection of observations.

Raises

ValueError – If evaluation or test dataset are empty.

ncem.estimators.EstimatorEDncem.split_data_target_cell

EstimatorEDncem.**split_data_target_cell**(*target_cell*, *test_split*, *validation_split*, *seed=1*)

Split nodes randomly into partitions.

Parameters

- **target_cell** (*str*) – Target cell type.
- **test_split** (*float*) – Fraction of total nodes to be in test set.
- **validation_split** (*float*) – Fraction of train-eval nodes to be in validation split.
- **seed** (*int*) – Seed for random selection of observations.

Raises

ValueError – If evaluation or test dataset are empty.

ncem.estimators.EstimatorEDncem.train

EstimatorEDncem.**train**(*epochs=1000*, *epochs_warmup=0*, *max_steps_per_epoch=20*, *batch_size=16*, *validation_batch_size=16*, *max_validation_steps=10*, *shuffle_buffer_size=10000*, *patience=20*, *lr_schedule_min_lr=1e-05*, *lr_schedule_factor=0.2*, *lr_schedule_patience=5*, *initial_epoch=0*, *monitor_partition='val'*, *monitor_metric='loss'*, *log_dir=None*, *callbacks=None*, *early_stopping=True*, *reduce_lr_plateau=True*, *pretrain_decoder=False*, *decoder_epochs=1000*, *decoder_patience=20*, *decoder_callbacks=None*, *aggressive=False*, *aggressive_enc_patience=10*, *aggressive_epochs=5*, *seed=1234*, ***kwargs*)

Train model.

Use validation loss and maximum number of epochs as termination criteria.

Parameters

- **epochs** (*int*) – Integer number of times to iterate over the training data arrays. If unspecified, it will default to 1000.
- **epochs_warmup** (*int*) – Integer number of times to iterate over the training data arrays in warm up (without early stopping). If unspecified, it will default to 0.
- **max_steps_per_epoch** (*int, optional*) – Maximal steps per epoch. If unspecified, it will default to 20.
- **batch_size** (*int*) – Number of samples per gradient update. If unspecified, it will default to 16.
- **validation_batch_size** (*int*) – Number of samples in validation. If unspecified, it will default to 16.
- **max_validation_steps** (*int*) – Maximal steps per validation. If unspecified, it will default to 10.
- **shuffle_buffer_size** (*int, optional*) – Shuffle buffer size. If unspecified, it will default to 1e4.
- **patience** (*int*) – Number of epochs with no improvement. If unspecified, it will default to 20.
- **lr_schedule_min_lr** (*float*) – Lower bound on the learning rate. If unspecified, it will default to 1e-5.
- **lr_schedule_factor** (*float*) – Factor by which the learning rate will be reduced. new_lr = lr * factor. If unspecified, it will default to 0.2.
- **lr_schedule_patience** (*int*) – Number of epochs with no improvement after which learning rate will be reduced. If unspecified, it will default to 5.
- **initial_epoch** (*int*) – Epoch at which to start training (useful for resuming a previous training run). If unspecified, it will default to 0.
- **monitor_partition** (*str*) – Monitor partition.
- **monitor_metric** (*str*) – Monitor metric.
- **log_dir** (*str, optional*) – Logging directory.
- **callbacks** (*list, optional*) – List of callbacks to be called during training.

- **early_stopping** (*bool*) – Whether to activate early stopping.
- **reduce_lr_plateau** (*bool*) – Whether to reduce learning rate on plateau.
- **pretrain_decoder** (*bool*) – Whether to pretrain the decoder model.
- **decoder_epochs** (*int*) – Integer number of times to iterate over the training data arrays in decoder pretraining. If unspecified, it will default to 1000.
- **decoder_patience** (*int*) – Number of epochs with no improvement in decoder pretraining. If unspecified, it will default to 20.
- **decoder_callbacks** (*list, optional*) – List of callbacks to be called during decoder pretraining.
- **aggressive** (*bool*) – Whether to train aggressive.
- **aggressive_enc_patience** (*int*) – Number of epochs with no improvement in aggressive training. If unspecified, it will default to 10.
- **aggressive_epochs** (*int*) – Integer number of times to iterate over the training data arrays in aggressive training. If unspecified, it will default to 5.
- **seed** (*int*) – Random seed for reproduability.
- **kwargs** – Arbitrary keyword arguments.

ncem.estimators.EstimatorEDncem.train_aggressive

EstimatorEDncem.**train_aggressive**(*aggressive_enc_patience=10*, *aggressive_epochs=5*)

Train model aggressive.

Parameters

- **aggressive_enc_patience** (*int*) – Number of epochs with no improvement in aggressive training. If unspecified, it will default to 10.
- **aggressive_epochs** (*int*) – Integer number of times to iterate over the training data arrays in aggressive training. If unspecified, it will default to 5.

ncem.estimators.EstimatorEDncem.train_normal

EstimatorEDncem.**train_normal**(*epochs=1000*, *patience=20*, *lr_schedule_min_lr=1e-05*, *lr_schedule_factor=0.2*, *lr_schedule_patience=5*, *initial_epoch=0*, *monitor_partition='val'*, *monitor_metric='loss'*, *log_dir=None*, *callbacks=None*, *early_stopping=True*, *reduce_lr_plateau=True*, ***kwargs*)

Train model normal.

Use validation loss and maximum number of epochs as termination criteria.

Parameters

- **epochs** (*int*) – Integer number of times to iterate over the training data arrays. If unspecified, it will default to 1000.
- **patience** (*int*) – Number of epochs with no improvement. If unspecified, it will default to 20.
- **lr_schedule_min_lr** (*float*) – Lower bound on the learning rate. If unspecified, it will default to 1e-5.

- **lr_schedule_factor** (*float*) – Factor by which the learning rate will be reduced. new_lr = lr * factor. If unspecified, it will default to 0.2.
- **lr_schedule_patience** (*int*) – Number of epochs with no improvement after which learning rate will be reduced. If unspecified, it will default to 5.
- **initial_epoch** (*int*) – Epoch at which to start training (useful for resuming a previous training run). If unspecified, it will default to 0.
- **monitor_partition** (*str*) – Monitor partition.
- **monitor_metric** (*str*) – Monitor metric.
- **log_dir** (*str, optional*) – Logging directory.
- **callbacks** (*list, optional*) – List of callbacks to be called during training.
- **early_stopping** (*bool*) – Whether to activate early stopping.
- **reduce_lr_plateau** (*bool*) – Whether to reduce learning rate on plateau.
- **kwargs** – Arbitrary keyword arguments.

Parameters

- **cond_type** (*str*) –
- **use_type_cond** (*bool*) –
- **log_transform** (*bool*) –

3.1.8 ncem.estimators.EstimatorInteractions

class ncem.estimators.**EstimatorInteractions**(*log_transform=False*)

Estimator class for interactions models. Subclass of Estimator.

Attributes

img_keys_all	Return all image keys.
nodes_idx_all	Return all node indices.
nodes_idx_eval	alias of Dict[str, list]
nodes_idx_test	alias of Dict[str, list]
nodes_idx_train	alias of Dict[str, list]
patient_ids_bytarget	Return patient identifiers by target.
patient_ids_unique	Return unique patient identifiers.

ncem.estimators.EstimatorInteractions.img_keys_all

property EstimatorInteractions.**img_keys_all**

Return all image keys.

Return type
img_keys_all

ncem.estimators.EstimatorInteractions.nodes_idx_all

property EstimatorInteractions.**nodes_idx_all**

Return all node indices.

Return type
nodes_idx_all

ncem.estimators.EstimatorInteractions.nodes_idx_eval

EstimatorInteractions.**nodes_idx_eval**

alias of Dict[str, list]

alias of Dict[str, list] .. autoattribute:: EstimatorInteractions.nodes_idx_eval

ncem.estimators.EstimatorInteractions.nodes_idx_test

EstimatorInteractions.**nodes_idx_test**

alias of Dict[str, list]

alias of Dict[str, list] .. autoattribute:: EstimatorInteractions.nodes_idx_test

ncem.estimators.EstimatorInteractions.nodes_idx_train

EstimatorInteractions.**nodes_idx_train**

alias of Dict[str, list]

alias of Dict[str, list] .. autoattribute:: EstimatorInteractions.nodes_idx_train

ncem.estimators.EstimatorInteractions.patient_ids_bytarget

property EstimatorInteractions.**patient_ids_bytarget: ndarray**

Return patient identifiers by target.

Return type
patient_ids_bytarget

ncem.estimators.EstimatorInteractions.patient_ids_unique

property EstimatorInteractions.**patient_ids_unique: ndarray**

Return unique patient identifiers.

Return type
patient_ids_unique

Methods

evaluate_any(img_keys, node_idx[, batch_size])	Evaluate model on any given data set.
evaluate_per_node_type([batch_size])	Evaluate model for each node type seperately.
get_data(data_origin, data_path, radius[, ...])	Get data used in estimator classes.
init_model([optimizer, learning_rate, ...])	Initialize a ModelInteractions object.
predict([batch_size])	Return observed labels and full predictions (including scale model) grouped exactly as in nodes_idx_test.
pretrain_decoder([decoder_epochs, patience, ...])	Pre-train decoder model.
split_data_given(img_keys_test, ...)	Split data by given partition.
split_data_node(test_split, validation_split)	Split nodes randomly into partitions.
split_data_target_cell(target_cell, ...[, seed])	Split nodes randomly into partitions.
train([epochs, epochs_warmup, ...])	Train model.
train_aggressive([aggressive_enc_patience, ...])	Train model aggressive.
train_normal([epochs, patience, ...])	Train model normal.

ncem.estimators.EstimatorInteractions.evaluate_any

EstimatorInteractions.**evaluate_any**(*img_keys*, *node_idx*, *batch_size=1*)

Evaluate model on any given data set.

Parameters

- **img_keys** – Image keys.
- **node_idx** – Nodes indices.
- **batch_size** (*int*) – Number of samples. If unspecified, it will default to 1.

Return type
eval_dict

ncem.estimators.EstimatorInteractions.evaluate_per_node_type

EstimatorInteractions.**evaluate_per_node_type**(*batch_size=1*)

Evaluate model for each node type seperately.

Parameters
batch_size (*int*) – Number of samples. If unspecified, it will default to 1.

Return type
split_per_node_type, evaluation_per_node_type

ncem.estimators.EstimatorInteractions.get_data

EstimatorInteractions.**get_data**(*data_origin*, *data_path*, *radius*, *n_rings=1*, *graph_covar_selection=None*, *node_label_space_id='type'*, *node_feature_space_id='standard'*, *use_covar_node_position=False*, *use_covar_node_label=False*, *use_covar_graph_covar=False*, *domain_type='image'*, *robustness=None*, *robustness_seed=1*, *n_top_genes=None*, *segmentation_robustness=None*, *resimulate_nodes=False*, *resimulate_nodes_w_depdency=False*, *resimulate_nodes_sparsity_rate=0.5*)

Get data used in estimator classes.

Parameters

- **data_origin** (*str*) – Data origin.
- **data_path** (*str*) – Data path.
- **radius** (*int, optional*) – Radius.
- **n_rings** (*int*) – Number of rings of neighbors for grid data.
- **graph_covar_selection** (*list, tuple, optional*) – Selected graph covariates.
- **node_label_space_id** (*str*) – Node label space id.
- **node_feature_space_id** (*str*) – Node feature space id.
- **use_covar_node_position** (*bool*) – Whether to use node position as covariate.
- **use_covar_node_label** (*bool*) – Whether to use node label as covariate.
- **use_covar_graph_covar** (*bool*) – Whether to use graph covariates.
- **domain_type** (*str*) – Covariate that is used as domain.
- **robustness** (*float, optional*) – Optional fraction of images for robustness test.
- **robustness_seed** (*int*) – Seed for robustness analysis
- **n_top_genes** (*int, optional*) – N top genes for highly variable gene selection.
- **segmentation_robustness** (*list, optional*) – Parameters for segmentation robustness fit, float for fraction of nodes and float for signal overflow.
- **resimulate_nodes** (*bool*) –
- **resimulate_nodes_w_depdency** (*bool*) –
- **resimulate_nodes_sparsity_rate** (*float*) –

Raises

ValueError – If sub-selected covar_selection could not be found, *node_label_space_id* or *node_feature_space_id* not recognized

ncem.estimators.EstimatorInteractions.init_model

EstimatorInteractions.**init_model**(*optimizer='adam'*, *learning_rate=0.0001*, *n_eval_nodes_per_graph=32*, *l2_coef=0.0*, *l1_coef=0.0*, *use_interactions=True*, *use_domain=False*, *scale_node_size=False*, *output_layer='linear'*, ***kwargs*)

Initialize a ModelInteractions object.

Parameters

- **optimizer** (*str*) – Optimizer.
- **learning_rate** (*float*) – Learning rate.
- **l2_coef** (*float*) – l2 regularization coefficient.
- **l1_coef** (*float*) – l1 regularization coefficient.
- **n_eval_nodes_per_graph** (*int*) – Number of nodes per graph.
- **use_interactions** (*bool*) – Whether to use source type.
- **use_domain** (*bool*) – Whether to use domain information.
- **scale_node_size** (*bool*) – Whether to scale output layer by node sizes.
- **output_layer** (*str*) – Output layer.
- **kwargs** – Arbitrary keyword arguments.

ncem.estimators.EstimatorInteractions.predict

EstimatorInteractions.**predict**(*batch_size=1*)

Return observed labels and full predictions (including scale model) grouped exactly as in nodes_idx_test.

Parameters

batch_size (*int*) – Number of samples. If unspecified, it will default to 1.

Return type

predict

ncem.estimators.EstimatorInteractions.pretrain_decoder

EstimatorInteractions.**pretrain_decoder**(*decoder_epochs=1000*, *patience=20*, *lr_schedule_min_lr=1e-05*, *lr_schedule_factor=0.2*, *lr_schedule_patience=5*, *initial_epoch=0*, *monitor_partition='val'*, *monitor_metric='loss'*, *log_dir=None*, *callbacks=None*, *early_stopping=True*, *reduce_lr_plateau=True*, ***kwargs*)

Pre-train decoder model.

Use validation loss and maximum number of epochs as termination criteria.

Parameters

- **patience** (*int*) – Number of epochs with no improvement. If unspecified, it will default to 20.
- **lr_schedule_min_lr** (*float*) – Lower bound on the learning rate. If unspecified, it will default to 1e-5.

- **lr_schedule_factor** (*float*) – Factor by which the learning rate will be reduced. new_lr = lr * factor. If unspecified, it will default to 0.2.
- **lr_schedule_patience** (*int*) – Number of epochs with no improvement after which learning rate will be reduced. If unspecified, it will default to 5.
- **initial_epoch** (*int*) – Epoch at which to start training (useful for resuming a previous training run). If unspecified, it will default to 0.
- **monitor_partition** (*str*) – Monitor partition.
- **monitor_metric** (*str*) – Monitor metric.
- **log_dir** (*str, optional*) – Logging directory.
- **callbacks** (*list, optional*) – List of callbacks to be called during training.
- **early_stopping** (*bool*) – Whether to activate early stopping.
- **reduce_lr_plateau** (*bool*) – Whether to reduce learning rate on plateau.
- **decoder_epochs** (*int*) – Integer number of times to iterate over the training data arrays in decoder pretraining. If unspecified, it will default to 1000.
- **kwargs** – Arbitrary keyword arguments.

ncem.estimators.EstimatorInteractions.split_data_given

EstimatorInteractions.**split_data_given**(*img_keys_test*, *img_keys_train*, *img_keys_eval*, *nodes_idx_test*, *nodes_idx_train*, *nodes_idx_eval*)

Split data by given partition.

Parameters

- **img_keys_test** – Test image keys.
- **img_keys_train** – Train image keys.
- **img_keys_eval** – Evaluation image keys.
- **nodes_idx_test** – Test node indices.
- **nodes_idx_train** – Train node indices.
- **nodes_idx_eval** – Evaluation node indices.

ncem.estimators.EstimatorInteractions.split_data_node

EstimatorInteractions.**split_data_node**(*test_split*, *validation_split*, *seed=1*)

Split nodes randomly into partitions.

Parameters

- **test_split** (*float*) – Fraction of total nodes to be in test set.
- **validation_split** (*float*) – Fraction of train-eval nodes to be in validation split.
- **seed** (*int*) – Seed for random selection of observations.

Raises

ValueError – If evaluation or test dataset are empty.

ncem.estimators.EstimatorInteractions.split_data_target_cell

EstimatorInteractions.**split_data_target_cell**(*target_cell*, *test_split*, *validation_split*, *seed=1*)

Split nodes randomly into partitions.

Parameters

- **target_cell** (*str*) – Target cell type.
- **test_split** (*float*) – Fraction of total nodes to be in test set.
- **validation_split** (*float*) – Fraction of train-eval nodes to be in validation split.
- **seed** (*int*) – Seed for random selection of observations.

Raises

ValueError – If evaluation or test dataset are empty.

ncem.estimators.EstimatorInteractions.train

EstimatorInteractions.**train**(*epochs=1000*, *epochs_warmup=0*, *max_steps_per_epoch=20*, *batch_size=16*, *validation_batch_size=16*, *max_validation_steps=10*, *shuffle_buffer_size=10000*, *patience=20*, *lr_schedule_min_lr=1e-05*, *lr_schedule_factor=0.2*, *lr_schedule_patience=5*, *initial_epoch=0*, *monitor_partition='val'*, *monitor_metric='loss'*, *log_dir=None*, *callbacks=None*, *early_stopping=True*, *reduce_lr_plateau=True*, *pretrain_decoder=False*, *decoder_epochs=1000*, *decoder_patience=20*, *decoder_callbacks=None*, *aggressive=False*, *aggressive_enc_patience=10*, *aggressive_epochs=5*, *seed=1234*, ***kwargs*)

Train model.

Use validation loss and maximum number of epochs as termination criteria.

Parameters

- **epochs** (*int*) – Integer number of times to iterate over the training data arrays. If unspecified, it will default to 1000.
- **epochs_warmup** (*int*) – Integer number of times to iterate over the training data arrays in warm up (without early stopping). If unspecified, it will default to 0.
- **max_steps_per_epoch** (*int, optional*) – Maximal steps per epoch. If unspecified, it will default to 20.
- **batch_size** (*int*) – Number of samples per gradient update. If unspecified, it will default to 16.
- **validation_batch_size** (*int*) – Number of samples in validation. If unspecified, it will default to 16.
- **max_validation_steps** (*int*) – Maximal steps per validation. If unspecified, it will default to 10.
- **shuffle_buffer_size** (*int, optional*) – Shuffle buffer size. If unspecified, it will default to 1e4.
- **patience** (*int*) – Number of epochs with no improvement. If unspecified, it will default to 20.

- **lr_schedule_min_lr** (*float*) – Lower bound on the learning rate. If unspecified, it will default to 1e-5.
- **lr_schedule_factor** (*float*) – Factor by which the learning rate will be reduced. new_lr = lr * factor. If unspecified, it will default to 0.2.
- **lr_schedule_patience** (*int*) – Number of epochs with no improvement after which learning rate will be reduced. If unspecified, it will default to 5.
- **initial_epoch** (*int*) – Epoch at which to start training (useful for resuming a previous training run). If unspecified, it will default to 0.
- **monitor_partition** (*str*) – Monitor partition.
- **monitor_metric** (*str*) – Monitor metric.
- **log_dir** (*str, optional*) – Logging directory.
- **callbacks** (*list, optional*) – List of callbacks to be called during training.
- **early_stopping** (*bool*) – Whether to activate early stopping.
- **reduce_lr_plateau** (*bool*) – Whether to reduce learning rate on plateau.
- **pretrain_decoder** (*bool*) – Whether to pretrain the decoder model.
- **decoder_epochs** (*int*) – Integer number of times to iterate over the training data arrays in decoder pretraining. If unspecified, it will default to 1000.
- **decoder_patience** (*int*) – Number of epochs with no improvement in decoder pretraining. If unspecified, it will default to 20.
- **decoder_callbacks** (*list, optional*) – List of callbacks to be called during decoder pretraining.
- **aggressive** (*bool*) – Whether to train aggressive.
- **aggressive_enc_patience** (*int*) – Number of epochs with no improvement in aggressive training. If unspecified, it will default to 10.
- **aggressive_epochs** (*int*) – Integer number of times to iterate over the training data arrays in aggressive training. If unspecified, it will default to 5.
- **seed** (*int*) – Random seed for reproduability.
- **kwargs** – Arbitrary keyword arguments.

ncem.estimators.EstimatorInteractions.train_aggressive

EstimatorInteractions.**train_aggressive**(*aggressive_enc_patience=10*, *aggressive_epochs=5*)

Train model aggressive.

Parameters

- **aggressive_enc_patience** (*int*) – Number of epochs with no improvement in aggressive training. If unspecified, it will default to 10.
- **aggressive_epochs** (*int*) – Integer number of times to iterate over the training data arrays in aggressive training. If unspecified, it will default to 5.

ncem.estimators.EstimatorInteractions.train_normal

EstimatorInteractions.**train_normal**(*epochs=1000*, *patience=20*, *lr_schedule_min_lr=1e-05*, *lr_schedule_factor=0.2*, *lr_schedule_patience=5*, *initial_epoch=0*, *monitor_partition='val'*, *monitor_metric='loss'*, *log_dir=None*, *callbacks=None*, *early_stopping=True*, *reduce_lr_plateau=True*, ***kwargs*)

Train model normal.

Use validation loss and maximum number of epochs as termination criteria.

Parameters

- **epochs** (*int*) – Integer number of times to iterate over the training data arrays. If unspecified, it will default to 1000.
- **patience** (*int*) – Number of epochs with no improvement. If unspecified, it will default to 20.
- **lr_schedule_min_lr** (*float*) – Lower bound on the learning rate. If unspecified, it will default to 1e-5.
- **lr_schedule_factor** (*float*) – Factor by which the learning rate will be reduced. new_lr = lr * factor. If unspecified, it will default to 0.2.
- **lr_schedule_patience** (*int*) – Number of epochs with no improvement after which learning rate will be reduced. If unspecified, it will default to 5.
- **initial_epoch** (*int*) – Epoch at which to start training (useful for resuming a previous training run). If unspecified, it will default to 0.
- **monitor_partition** (*str*) – Monitor partition.
- **monitor_metric** (*str*) – Monitor metric.
- **log_dir** (*str, optional*) – Logging directory.
- **callbacks** (*list, optional*) – List of callbacks to be called during training.
- **early_stopping** (*bool*) – Whether to activate early stopping.
- **reduce_lr_plateau** (*bool*) – Whether to reduce learning rate on plateau.
- **kwargs** – Arbitrary keyword arguments.

Parameters

log_transform (*bool*) –

3.1.9 ncem.estimators.EstimatorLinear

class ncem.estimators.**EstimatorLinear**(*log_transform=False*)

Estimator class for linear models. Subclass of Estimator.

Attributes

img_keys_all	Return all image keys.
nodes_idx_all	Return all node indices.
nodes_idx_eval	alias of Dict[str, list]
nodes_idx_test	alias of Dict[str, list]
nodes_idx_train	alias of Dict[str, list]
patient_ids_bytarget	Return patient identifiers by target.
patient_ids_unique	Return unique patient identifiers.

ncem.estimators.EstimatorLinear.img_keys_all

property EstimatorLinear.**img_keys_all**

Return all image keys.

Return type
img_keys_all

ncem.estimators.EstimatorLinear.nodes_idx_all

property EstimatorLinear.**nodes_idx_all**

Return all node indices.

Return type
nodes_idx_all

ncem.estimators.EstimatorLinear.nodes_idx_eval

EstimatorLinear.**nodes_idx_eval**

alias of Dict[str, list]

alias of Dict[str, list] .. autoattribute:: EstimatorLinear.nodes_idx_eval

ncem.estimators.EstimatorLinear.nodes_idx_test

EstimatorLinear.**nodes_idx_test**

alias of Dict[str, list]

alias of Dict[str, list] .. autoattribute:: EstimatorLinear.nodes_idx_test

ncem.estimators.EstimatorLinear.nodes_idx_train

EstimatorLinear.**nodes_idx_train**

alias of Dict[str, list]

alias of Dict[str, list] .. autoattribute:: EstimatorLinear.nodes_idx_train

ncem.estimators.EstimatorLinear.patient_ids_bytarget

property EstimatorLinear.**patient_ids_bytarget: ndarray**

Return patient identifiers by target.

Return type
patient_ids_bytarget

ncem.estimators.EstimatorLinear.patient_ids_unique

property EstimatorLinear.**patient_ids_unique: ndarray**

Return unique patient identifiers.

Return type
patient_ids_unique

Methods

evaluate_any(img_keys, node_idx[, batch_size])	Evaluate model on any given data set.
evaluate_per_node_type([batch_size])	Evaluate model for each node type seperately.
get_data(data_origin, data_path, radius[, ...])	Get data used in estimator classes.
init_model([optimizer, learning_rate, ...])	Initialize a ModelInteractions object.
predict([batch_size])	Return observed labels and full predictions (including scale model) grouped exactly as in nodes_idx_test.
pretrain_decoder([decoder_epochs, patience, ...])	Pre-train decoder model.
split_data_given(img_keys_test, ...)	Split data by given partition.
split_data_node(test_split, validation_split)	Split nodes randomly into partitions.
split_data_target_cell(target_cell, ...[, seed])	Split nodes randomly into partitions.
train([epochs, epochs_warmup, ...])	Train model.
train_aggressive([aggressive_enc_patience, ...])	Train model aggressive.
train_normal([epochs, patience, ...])	Train model normal.

ncem.estimators.EstimatorLinear.evaluate_any

EstimatorLinear.**evaluate_any**(*img_keys*, *node_idx*, *batch_size=1*)

Evaluate model on any given data set.

Parameters

- **img_keys** – Image keys.
- **node_idx** – Nodes indices.
- **batch_size** (*int*) – Number of samples. If unspecified, it will default to 1.

Return type
eval_dict

ncem.estimators.EstimatorLinear.evaluate_per_node_type

EstimatorLinear.**evaluate_per_node_type**(*batch_size=1*)

Evaluate model for each node type seperately.

Parameters

batch_size (*int*) – Number of samples. If unspecified, it will default to 1.

Return type

split_per_node_type, evaluation_per_node_type

ncem.estimators.EstimatorLinear.get_data

EstimatorLinear.**get_data**(*data_origin*, *data_path*, *radius*, *n_rings=1*, *graph_covar_selection=None*, *node_label_space_id='type'*, *node_feature_space_id='standard'*, *use_covar_node_position=False*, *use_covar_node_label=False*, *use_covar_graph_covar=False*, *domain_type='image'*, *robustness=None*, *robustness_seed=1*, *n_top_genes=None*, *segmentation_robustness=None*, *resimulate_nodes=False*, *resimulate_nodes_w_depdency=False*, *resimulate_nodes_sparsity_rate=0.5*)

Get data used in estimator classes.

Parameters

- **data_origin** (*str*) – Data origin.
- **data_path** (*str*) – Data path.
- **radius** (*int, optional*) – Radius.
- **n_rings** (*int*) – Number of rings of neighbors for grid data.
- **graph_covar_selection** (*list, tuple, optional*) – Selected graph covariates.
- **node_label_space_id** (*str*) – Node label space id.
- **node_feature_space_id** (*str*) – Node feature space id.
- **use_covar_node_position** (*bool*) – Whether to use node position as covariate.
- **use_covar_node_label** (*bool*) – Whether to use node label as covariate.
- **use_covar_graph_covar** (*bool*) – Whether to use graph covariates.
- **domain_type** (*str*) – Covariate that is used as domain.
- **robustness** (*float, optional*) – Optional fraction of images for robustness test.
- **robustness_seed** (*int*) – Seed for robustness analysis
- **n_top_genes** (*int, optional*) – N top genes for highly variable gene selection.
- **segmentation_robustness** (*list, optional*) – Parameters for segmentation robustness fit, float for fraction of nodes and float for signal overflow.
- **resimulate_nodes** (*bool*) –
- **resimulate_nodes_w_depdency** (*bool*) –
- **resimulate_nodes_sparsity_rate** (*float*) –

Raises

ValueError – If sub-selected covar_selection could not be found, *node_label_space_id* or *node_feature_space_id* not recognized

ncem.estimators.EstimatorLinear.init_model

EstimatorLinear.**init_model**(*optimizer='adam'*, *learning_rate=0.0001*, *n_eval_nodes_per_graph=32*, *l2_coef=0.0*, *l1_coef=0.0*, *use_source_type=True*, *use_domain=False*, *scale_node_size=False*, *output_layer='linear'*, ***kwargs*)

Initialize a ModelInteractions object.

Parameters

- **optimizer** (*str*) – Optimizer.
- **learning_rate** (*float*) – Learning rate.
- **l2_coef** (*float*) – l2 regularization coefficient.
- **l1_coef** (*float*) – l1 regularization coefficient.
- **n_eval_nodes_per_graph** (*int*) – Number of nodes per graph.
- **use_source_type** (*bool*) – Whether to use source type.
- **use_domain** (*bool*) – Whether to use domain information.
- **scale_node_size** (*bool*) – Whether to scale output layer by node sizes.
- **output_layer** (*str*) – Output layer.
- **kwargs** – Arbitrary keyword arguments.

ncem.estimators.EstimatorLinear.predict

EstimatorLinear.**predict**(*batch_size=1*)

Return observed labels and full predictions (including scale model) grouped exactly as in nodes_idx_test.

Parameters
batch_size (*int*) – Number of samples. If unspecified, it will default to 1.

Return type
predict

ncem.estimators.EstimatorLinear.pretrain_decoder

EstimatorLinear.**pretrain_decoder**(*decoder_epochs=1000*, *patience=20*, *lr_schedule_min_lr=1e-05*, *lr_schedule_factor=0.2*, *lr_schedule_patience=5*, *initial_epoch=0*, *monitor_partition='val'*, *monitor_metric='loss'*, *log_dir=None*, *callbacks=None*, *early_stopping=True*, *reduce_lr_plateau=True*, ***kwargs*)

Pre-train decoder model.

Use validation loss and maximum number of epochs as termination criteria.

Parameters

- **patience** (*int*) – Number of epochs with no improvement. If unspecified, it will default to 20.
- **lr_schedule_min_lr** (*float*) – Lower bound on the learning rate. If unspecified, it will default to 1e-5.
- **lr_schedule_factor** (*float*) – Factor by which the learning rate will be reduced. new_lr = lr * factor. If unspecified, it will default to 0.2.

- **lr_schedule_patience** (*int*) – Number of epochs with no improvement after which learning rate will be reduced. If unspecified, it will default to 5.
- **initial_epoch** (*int*) – Epoch at which to start training (useful for resuming a previous training run). If unspecified, it will default to 0.
- **monitor_partition** (*str*) – Monitor partition.
- **monitor_metric** (*str*) – Monitor metric.
- **log_dir** (*str, optional*) – Logging directory.
- **callbacks** (*list, optional*) – List of callbacks to be called during training.
- **early_stopping** (*bool*) – Whether to activate early stopping.
- **reduce_lr_plateau** (*bool*) – Whether to reduce learning rate on plateau.
- **decoder_epochs** (*int*) – Integer number of times to iterate over the training data arrays in decoder pretraining. If unspecified, it will default to 1000.
- **kwargs** – Arbitrary keyword arguments.

ncem.estimators.EstimatorLinear.split_data_given

EstimatorLinear.**split_data_given**(*img_keys_test*, *img_keys_train*, *img_keys_eval*, *nodes_idx_test*, *nodes_idx_train*, *nodes_idx_eval*)

Split data by given partition.

Parameters

- **img_keys_test** – Test image keys.
- **img_keys_train** – Train image keys.
- **img_keys_eval** – Evaluation image keys.
- **nodes_idx_test** – Test node indices.
- **nodes_idx_train** – Train node indices.
- **nodes_idx_eval** – Evaluation node indices.

ncem.estimators.EstimatorLinear.split_data_node

EstimatorLinear.**split_data_node**(*test_split*, *validation_split*, *seed=1*)

Split nodes randomly into partitions.

Parameters

- **test_split** (*float*) – Fraction of total nodes to be in test set.
- **validation_split** (*float*) – Fraction of train-eval nodes to be in validation split.
- **seed** (*int*) – Seed for random selection of observations.

Raises

ValueError – If evaluation or test dataset are empty.

ncem.estimators.EstimatorLinear.split_data_target_cell

EstimatorLinear.**split_data_target_cell**(*target_cell*, *test_split*, *validation_split*, *seed=1*)

Split nodes randomly into partitions.

Parameters

- **target_cell** (*str*) – Target cell type.
- **test_split** (*float*) – Fraction of total nodes to be in test set.
- **validation_split** (*float*) – Fraction of train-eval nodes to be in validation split.
- **seed** (*int*) – Seed for random selection of observations.

Raises

ValueError – If evaluation or test dataset are empty.

ncem.estimators.EstimatorLinear.train

EstimatorLinear.**train**(*epochs=1000*, *epochs_warmup=0*, *max_steps_per_epoch=20*, *batch_size=16*, *validation_batch_size=16*, *max_validation_steps=10*, *shuffle_buffer_size=10000*, *patience=20*, *lr_schedule_min_lr=1e-05*, *lr_schedule_factor=0.2*, *lr_schedule_patience=5*, *initial_epoch=0*, *monitor_partition='val'*, *monitor_metric='loss'*, *log_dir=None*, *callbacks=None*, *early_stopping=True*, *reduce_lr_plateau=True*, *pretrain_decoder=False*, *decoder_epochs=1000*, *decoder_patience=20*, *decoder_callbacks=None*, *aggressive=False*, *aggressive_enc_patience=10*, *aggressive_epochs=5*, *seed=1234*, ***kwargs*)

Train model.

Use validation loss and maximum number of epochs as termination criteria.

Parameters

- **epochs** (*int*) – Integer number of times to iterate over the training data arrays. If unspecified, it will default to 1000.
- **epochs_warmup** (*int*) – Integer number of times to iterate over the training data arrays in warm up (without early stopping). If unspecified, it will default to 0.
- **max_steps_per_epoch** (*int, optional*) – Maximal steps per epoch. If unspecified, it will default to 20.
- **batch_size** (*int*) – Number of samples per gradient update. If unspecified, it will default to 16.
- **validation_batch_size** (*int*) – Number of samples in validation. If unspecified, it will default to 16.
- **max_validation_steps** (*int*) – Maximal steps per validation. If unspecified, it will default to 10.
- **shuffle_buffer_size** (*int, optional*) – Shuffle buffer size. If unspecified, it will default to 1e4.
- **patience** (*int*) – Number of epochs with no improvement. If unspecified, it will default to 20.
- **lr_schedule_min_lr** (*float*) – Lower bound on the learning rate. If unspecified, it will default to 1e-5.

- **lr_schedule_factor** (*float*) – Factor by which the learning rate will be reduced. new_lr = lr * factor. If unspecified, it will default to 0.2.
- **lr_schedule_patience** (*int*) – Number of epochs with no improvement after which learning rate will be reduced. If unspecified, it will default to 5.
- **initial_epoch** (*int*) – Epoch at which to start training (useful for resuming a previous training run). If unspecified, it will default to 0.
- **monitor_partition** (*str*) – Monitor partition.
- **monitor_metric** (*str*) – Monitor metric.
- **log_dir** (*str, optional*) – Logging directory.
- **callbacks** (*list, optional*) – List of callbacks to be called during training.
- **early_stopping** (*bool*) – Whether to activate early stopping.
- **reduce_lr_plateau** (*bool*) – Whether to reduce learning rate on plateau.
- **pretrain_decoder** (*bool*) – Whether to pretrain the decoder model.
- **decoder_epochs** (*int*) – Integer number of times to iterate over the training data arrays in decoder pretraining. If unspecified, it will default to 1000.
- **decoder_patience** (*int*) – Number of epochs with no improvement in decoder pretraining. If unspecified, it will default to 20.
- **decoder_callbacks** (*list, optional*) – List of callbacks to be called during decoder pretraining.
- **aggressive** (*bool*) – Whether to train aggressive.
- **aggressive_enc_patience** (*int*) – Number of epochs with no improvement in aggressive training. If unspecified, it will default to 10.
- **aggressive_epochs** (*int*) – Integer number of times to iterate over the training data arrays in aggressive training. If unspecified, it will default to 5.
- **seed** (*int*) – Random seed for reproduability.
- **kwargs** – Arbitrary keyword arguments.

ncem.estimators.EstimatorLinear.train_aggressive

EstimatorLinear.**train_aggressive**(*aggressive_enc_patience=10*, *aggressive_epochs=5*)

Train model aggressive.

Parameters

- **aggressive_enc_patience** (*int*) – Number of epochs with no improvement in aggressive training. If unspecified, it will default to 10.
- **aggressive_epochs** (*int*) – Integer number of times to iterate over the training data arrays in aggressive training. If unspecified, it will default to 5.

ncem.estimators.EstimatorLinear.train_normal

EstimatorLinear.**train_normal**(*epochs=1000*, *patience=20*, *lr_schedule_min_lr=1e-05*, *lr_schedule_factor=0.2*, *lr_schedule_patience=5*, *initial_epoch=0*, *monitor_partition='val'*, *monitor_metric='loss'*, *log_dir=None*, *callbacks=None*, *early_stopping=True*, *reduce_lr_plateau=True*, ***kwargs*)

Train model normal.

Use validation loss and maximum number of epochs as termination criteria.

Parameters

- **epochs** (*int*) – Integer number of times to iterate over the training data arrays. If unspecified, it will default to 1000.
- **patience** (*int*) – Number of epochs with no improvement. If unspecified, it will default to 20.
- **lr_schedule_min_lr** (*float*) – Lower bound on the learning rate. If unspecified, it will default to 1e-5.
- **lr_schedule_factor** (*float*) – Factor by which the learning rate will be reduced. new_lr = lr * factor. If unspecified, it will default to 0.2.
- **lr_schedule_patience** (*int*) – Number of epochs with no improvement after which learning rate will be reduced. If unspecified, it will default to 5.
- **initial_epoch** (*int*) – Epoch at which to start training (useful for resuming a previous training run). If unspecified, it will default to 0.
- **monitor_partition** (*str*) – Monitor partition.
- **monitor_metric** (*str*) – Monitor metric.
- **log_dir** (*str, optional*) – Logging directory.
- **callbacks** (*list, optional*) – List of callbacks to be called during training.
- **early_stopping** (*bool*) – Whether to activate early stopping.
- **reduce_lr_plateau** (*bool*) – Whether to reduce learning rate on plateau.
- **kwargs** – Arbitrary keyword arguments.

Parameters

log_transform (*bool*) –

3.2 Model classes: *models*

Model classes from ncem for advanced use.

Classes that wrap tensorflow models.

models.ModelCVAE(input_shapes[, latent_dim, ...])	Model class for conditional variational autoencoder.
models.ModelCVAEncem(input_shapes[, ...])	Model class for NCEM conditional variational autoencoder with graph layer IND (MAX) or GCN.
models.ModelED(input_shapes[, latent_dim, ...])	Model class for non-spatial encoder-decoder.
models.ModelEDncem(input_shapes[, ...])	Model class for NCEM encoder-decoder with graph layer IND (MAX) or GCN.
models.ModelInteractions(input_shapes[, ...])	Model class for interaction model, baseline and spatial model.
models.ModelLinear(input_shapes[, l2_coef, ...])	Model class for linear model, baseline and spatial model.

3.2.1 ncem.models.ModelCVAE

class ncem.models.**ModelCVAE**(*input_shapes*, *latent_dim=10*, *intermediate_dim_enc=128*, *intermediate_dim_dec=128*, *depth_enc=1*, *depth_dec=1*, *dropout_rate=0.1*, *l2_coef=0.0*, *l1_coef=0.0*, *use_domain=False*, *use_type_cond=True*, *use_batch_norm=False*, *scale_node_size=False*, *transform_input=False*, *output_layer='gaussian'*, ***kwargs*)

Model class for conditional variational autoencoder.

Methods

Parameters

- **latent_dim** (*int*) –
- **intermediate_dim_enc** (*int*) –
- **intermediate_dim_dec** (*int*) –
- **depth_enc** (*int*) –
- **depth_dec** (*int*) –
- **dropout_rate** (*float*) –
- **l2_coef** (*float*) –
- **l1_coef** (*float*) –
- **use_domain** (*bool*) –
- **use_type_cond** (*bool*) –
- **use_batch_norm** (*bool*) –
- **scale_node_size** (*bool*) –
- **transform_input** (*bool*) –

3.2.2 ncem.models.ModelCVAEncem

class ncem.models.**ModelCVAEncem**(*input_shapes*, *latent_dim=10*, *intermediate_dim_enc=128*, *intermediate_dim_dec=128*, *depth_enc=1*, *depth_dec=1*, *dropout_rate=0.1*, *l2_coef=0.0*, *l1_coef=0.0*, *cond_type='gcn'*, *cond_depth=1*, *cond_dim=8*, *cond_dropout_rate=0.1*, *cond_activation='relu'*, *cond_l2_reg=0.0*, *cond_use_bias=True*, *use_domain=False*, *scale_node_size=False*, *use_type_cond=True*, *use_batch_norm=False*, *transform_input=False*, *output_layer='gaussian'*, ***kwargs*)

Model class for NCEM conditional variational autoencoder with graph layer IND (MAX) or GCN.

Methods

Parameters

- **latent_dim** (*int*) –
- **intermediate_dim_enc** (*int*) –
- **intermediate_dim_dec** (*int*) –
- **depth_enc** (*int*) –
- **depth_dec** (*int*) –
- **dropout_rate** (*float*) –
- **l2_coef** (*float*) –
- **l1_coef** (*float*) –
- **cond_type** (*str*) –
- **cond_depth** (*int*) –
- **cond_dim** (*int*) –
- **cond_dropout_rate** (*float*) –
- **cond_activation** (*Union[str, Layer]*) –
- **cond_l2_reg** (*float*) –
- **cond_use_bias** (*bool*) –
- **use_domain** (*bool*) –
- **scale_node_size** (*bool*) –
- **use_type_cond** (*bool*) –
- **use_batch_norm** (*bool*) –
- **transform_input** (*bool*) –
- **output_layer** (*str*) –

3.2.3 ncem.models.ModelED

class ncem.models.**ModelED**(*input_shapes*, *latent_dim=10*, *dropout_rate=0.1*, *l2_coef=0.0*, *l1_coef=0.0*, *enc_intermediate_dim=128*, *enc_depth=2*, *dec_intermediate_dim=128*, *dec_depth=2*, *use_domain=False*, *use_type_cond=True*, *scale_node_size=False*, *output_layer='gaussian'*, ***kwargs*)

Model class for non-spatial encoder-decoder.

Methods

Parameters

- **latent_dim** (*int*) –
- **dropout_rate** (*float*) –
- **l2_coef** (*float*) –
- **l1_coef** (*float*) –
- **enc_intermediate_dim** (*int*) –
- **enc_depth** (*int*) –
- **dec_intermediate_dim** (*int*) –
- **dec_depth** (*int*) –
- **use_domain** (*bool*) –
- **use_type_cond** (*bool*) –
- **scale_node_size** (*bool*) –
- **output_layer** (*str*) –

3.2.4 ncem.models.ModelEDncem

class ncem.models.**ModelEDncem**(*input_shapes*, *latent_dim=10*, *dropout_rate=0.1*, *l2_coef=0.0*, *l1_coef=0.0*, *enc_intermediate_dim=128*, *enc_depth=2*, *dec_intermediate_dim=128*, *dec_depth=2*, *cond_type='gcn'*, *cond_depth=1*, *cond_dim=8*, *cond_dropout_rate=0.1*, *cond_activation='relu'*, *cond_l2_reg=0.0*, *cond_use_bias=True*, *use_domain=False*, *use_type_cond=False*, *scale_node_size=False*, *output_layer='gaussian'*, ***kwargs*)

Model class for NCEM encoder-decoder with graph layer IND (MAX) or GCN.

Methods

Parameters

- **latent_dim** (*int*) –
- **dropout_rate** (*float*) –
- **l2_coef** (*float*) –
- **l1_coef** (*float*) –
- **enc_intermediate_dim** (*int*) –
- **enc_depth** (*int*) –
- **dec_intermediate_dim** (*int*) –
- **dec_depth** (*int*) –
- **cond_type** (*str*) –
- **cond_depth** (*int*) –
- **cond_dim** (*int*) –
- **cond_dropout_rate** (*float*) –
- **cond_activation** (*Union[str, Layer]*) –
- **cond_l2_reg** (*float*) –
- **cond_use_bias** (*bool*) –
- **use_domain** (*bool*) –
- **use_type_cond** (*bool*) –
- **scale_node_size** (*bool*) –
- **output_layer** (*str*) –

3.2.5 ncem.models.ModelInteractions

class ncem.models.**ModelInteractions**(*input_shapes*, *l2_coef=0.0*, *l1_coef=0.0*, *use_interactions=False*, *use_domain=False*, *scale_node_size=False*, *output_layer='linear'*, ***kwargs*)

Model class for interaction model, baseline and spatial model.

Methods

Parameters

- **l2_coef** (*Optional[float]*) –
- **l1_coef** (*Optional[float]*) –
- **use_interactions** (*bool*) –

- **use_domain** (*bool*) –
- **scale_node_size** (*bool*) –
- **output_layer** (*str*) –

3.2.6 ncem.models.ModelLinear

class ncem.models.**ModelLinear**(*input_shapes*, *l2_coef=0.0*, *l1_coef=0.0*, *use_source_type=False*, *use_domain=False*, *scale_node_size=False*, *output_layer='linear'*, ***kwargs*)

Model class for linear model, baseline and spatial model.

Attributes:
args (dict): training_model:

Raises:
ValueError: If *output_layer* is not recognized.

Methods

Parameters

- **l2_coef** (*float*) –
- **l1_coef** (*float*) –
- **use_source_type** (*bool*) –
- **use_domain** (*bool*) –
- **scale_node_size** (*bool*) –
- **output_layer** (*str*) –

3.3 Train: *train*

The interface for training ncem compatible models.

3.3.1 Trainer classes

Classes that wrap estimator classes to use in grid search training.

train.TrainModelCVAE()
train.TrainModelCVAEncem()
train.TrainModelED()
train.TrainModelEDncem()
train.TrainModelInteractions()
train.TrainModelLinear()

ncem.train.TrainModelCVAE

class ncem.train.**TrainModelCVAE**

Attributes

estimator

ncem.train.TrainModelCVAE.estimator

TrainModelCVAE.**estimator:** *EstimatorCVAE*

Methods

init_estim(**kwargs)	
save(fn[, save_weights])	Save weights and summary statistics.
save_time(fn, duration)	

ncem.train.TrainModelCVAE.init_estim

TrainModelCVAE.**init_estim**(***kwargs*)

ncem.train.TrainModelCVAE.save

TrainModelCVAE.**save**(*fn*, *save_weights=True*)

Save weights and summary statistics.

Parameters

- **fn** (*str*) –
- **save_weights** (*bool*) –

ncem.train.TrainModelCVAE.save_time

TrainModelCVAE.**save_time**(*fn*, *duration*)

Parameters

fn (*str*) –

ncem.train.TrainModelCVAEncem

class ncem.train.**TrainModelCVAEncem**

Attributes

estimator

ncem.train.TrainModelCVAEncem.estimator

TrainModelCVAEncem.**estimator:** *EstimatorCVAEncem*

Methods

init_estim(**kwargs)	
save(fn[, save_weights])	Save weights and summary statistics.
save_time(fn, duration)	

ncem.train.TrainModelCVAEncem.init_estim

TrainModelCVAEncem.**init_estim**(***kwargs*)

ncem.train.TrainModelCVAEncem.save

TrainModelCVAEncem.**save**(*fn*, *save_weights=True*)

Save weights and summary statistics.

Parameters

- **fn** (*str*) –
- **save_weights** (*bool*) –

ncem.train.TrainModelCVAEncem.save_time

TrainModelCVAEncem.**save_time**(*fn*, *duration*)

Parameters

fn (*str*) –

ncem.train.TrainModelED

class ncem.train.**TrainModelED**

Attributes

estimator

ncem.train.TrainModelED.estimator

TrainModelED.**estimator:** *EstimatorED*

Methods

init_estim(**kwargs)	
save(fn[, save_weights])	Save weights and summary statistics.
save_time(fn, duration)	

ncem.train.TrainModelED.init_estim

TrainModelED.**init_estim**(***kwargs*)

ncem.train.TrainModelED.save

TrainModelED.**save**(*fn*, *save_weights=True*)

Save weights and summary statistics.

Parameters

- **fn** (*str*) –
- **save_weights** (*bool*) –

ncem.train.TrainModelED.save_time

TrainModelED.**save_time**(*fn*, *duration*)

Parameters

fn (*str*) –

ncem.train.TrainModelEDncem

class ncem.train.**TrainModelEDncem**

Attributes

estimator

ncem.train.TrainModelEDncem.estimator

TrainModelEDncem.**estimator:** *EstimatorEDncem*

Methods

init_estim(**kwargs)	
save(fn[, save_weights])	Save weights and summary statistics.
save_time(fn, duration)	

ncem.train.TrainModelEDncem.init_estim

TrainModelEDncem.**init_estim**(***kwargs*)

ncem.train.TrainModelEDncem.save

TrainModelEDncem.**save**(*fn*, *save_weights=True*)

Save weights and summary statistics.

Parameters

- **fn** (*str*) –
- **save_weights** (*bool*) –

ncem.train.TrainModelEDncem.save_time

TrainModelEDncem.**save_time**(*fn*, *duration*)

Parameters

fn (*str*) –

ncem.train.TrainModelInteractions

class ncem.train.**TrainModelInteractions**

Attributes

estimator

ncem.train.TrainModelInteractions.estimator

TrainModelInteractions.**estimator:** *EstimatorInteractions*

Methods

init_estim(**kwargs)	
save(fn[, save_weights])	Save weights and summary statistics.
save_time(fn, duration)	

ncem.train.TrainModelInteractions.init_estim

TrainModelInteractions.**init_estim**(***kwargs*)

ncem.train.TrainModelInteractions.save

TrainModelInteractions.**save**(*fn*, *save_weights=True*)

Save weights and summary statistics.

Parameters

- **fn** (*str*) –
- **save_weights** (*bool*) –

ncem.train.TrainModelInteractions.save_time

TrainModelInteractions.**save_time**(*fn*, *duration*)

Parameters

fn (*str*) –

ncem.train.TrainModelLinear

class ncem.train.**TrainModelLinear**

Attributes

estimator

ncem.train.TrainModelLinear.estimator

TrainModelLinear.**estimator:** *EstimatorLinear*

Methods

init_estim(**kwargs)	
save(fn[, save_weights])	Save weights and summary statistics.
save_time(fn, duration)	

ncem.train.TrainModelLinear.init_estim

TrainModelLinear.**init_estim**(***kwargs*)

ncem.train.TrainModelLinear.save

TrainModelLinear.**save**(*fn*, *save_weights=True*)

Save weights and summary statistics.

Parameters

- **fn** (*str*) –
- **save_weights** (*bool*) –

ncem.train.TrainModelLinear.save_time

TrainModelLinear.**save_time**(*fn*, *duration*)

Parameters

fn (*str*) –

3.3.2 Grid search summaries

Classes to pool evaluation metrics across fits in a grid search.

train.GridSearchContainer(source_path, ...)	GridSearchContainer class.

ncem.train.GridSearchContainer

class ncem.train.**GridSearchContainer**(*source_path*, *gs_ids*, *lateral_resolution*)

GridSearchContainer class.

Attributes

cv_keys	Return keys of cross-validation used in dictionaries in this class.
runparams	
run_ids_clean	
source_gs	
cv_ids	
target_cell_runparams	
target_cell_evals	
target_cell_indices	

ncem.train.GridSearchContainer.cv_keys

property GridSearchContainer.**cv_keys: List[str]**

Return keys of cross-validation used in dictionaries in this class.

Return type
list of string keys

ncem.train.GridSearchContainer.runparams

GridSearchContainer.**runparams: dict**

ncem.train.GridSearchContainer.run_ids_clean

GridSearchContainer.**run_ids_clean: dict**

ncem.train.GridSearchContainer.source_gs

GridSearchContainer.**source_gs: dict**

ncem.train.GridSearchContainer.cv_ids

GridSearchContainer.**cv_ids: dict**

ncem.train.GridSearchContainer.target_cell_runparams

GridSearchContainer.**target_cell_runparams: dict**

ncem.train.GridSearchContainer.target_cell_evals

GridSearchContainer.**target_cell_evals: dict**

ncem.train.GridSearchContainer.target_cell_indices

GridSearchContainer.**target_cell_indices: dict**

Methods

copy_best_model(gs_id[, dst, metric_select, ...])	Copy best model.
get_best_model_id([subset_hyperparameters, ...])	Get best model identifier.
get_info(model_id[, expected_pickle])	Get information of model.
load_gs([expected_pickle, ...])	Load all metrics from grid search output files.
load_target_cell_evaluation([...])	Load all metrics from grid search output files of target cell evaluation.
plot_best_model_by_hyperparam(...[, ...])	Plot best model by hyperparameter.
plot_target_cell_evaluation(metric_show, ...)	Plot target cell evaluation.
select_cv(cv_idx)	Return key of of cross-validation selected with numeric index.

ncem.train.GridSearchContainer.copy_best_model

GridSearchContainer.**copy_best_model**(*gs_id*, *dst='best'*, *metric_select='loss'*, *partition_select='val'*, *cv_mode='mean'*)

Copy best model.

Parameters

- **gs_id** (*str*) – Grid search identifier.
- **dst** (*str*) – dst folder.
- **metric_select** (*str*) – Selected metric.
- **partition_select** (*str*) – Selected partition.
- **cv_mode** (*str*) – Cross validation mode.

ncem.train.GridSearchContainer.get_best_model_id

GridSearchContainer.**get_best_model_id**(*subset_hyperparameters=None*, *metric_select='r_squared_linreg'*, *partition_select='test'*, *cv_mode='mean'*)

Get best model identifier.

Parameters

- **subset_hyperparameters** (*list, optional*) – List of subset hyperparameters.
- **metric_select** (*str*) – Selected metric.
- **partition_select** (*str*) – Selected parition.
- **cv_mode** (*str*) – cross validation mode.

Return type
best_model_id

Raises

- **ValueError** – If measure, partition or cv_mode not recognized.
- **Warning** – If cv_mode max is selected with the following metrics: loss, elbo, mse, mae

ncem.train.GridSearchContainer.get_info

GridSearchContainer.**get_info**(*model_id*, *expected_pickle=None*)

Get information of model.

Parameters

- **model_id** – Model identifier.
- **expected_pickle** (*list, optional*) – Expected pickle files.

Raises
ValueError – If file is missing.

ncem.train.GridSearchContainer.load_gs

GridSearchContainer.**load_gs**(*expected_pickle=None*, *add_posterior_sampling_model=False*, *report_unsuccessful_runs=False*)

Load all metrics from grid search output files.

Core results are save in self.summary_table.

Parameters

- **expected_pickle** (*list, optional*) – Expected pickle files.
- **add_posterior_sampling_model** (*bool*) – Whether to add posterior sampling model as seperate model to summary_table.
- **report_unsuccessful_runs** (*bool*) – Whether to print reporting statements in out stream.

Raises
ValueError – If no complete runs found.

ncem.train.GridSearchContainer.load_target_cell_evaluation

GridSearchContainer.**load_target_cell_evaluation**(*report_unsuccessful_runs=False*)

Load all metrics from grid search output files of target cell evaluation.

Core results are save in self.target_cell_table.

Parameters

report_unsuccessful_runs (*bool*) – Whether to print reporting statements in out stream.

ncem.train.GridSearchContainer.plot_best_model_by_hyperparam

GridSearchContainer.**plot_best_model_by_hyperparam**(*graph_model_class*, *baseline_model_class*, *partition_show='test'*, *metric_show='r_squared_linreg'*, *partition_select='val'*, *metric_select='r_squared_linreg'*, *param_x='um_radius'*, *param_hue='model'*, *rename_levels=None*, *subset_hyperparameters=None*, *cv_mode='mean'*, *yaxis_limit=None*, *xticks=None*, *rotate_xticks=True*, *figsize=(3.5, 4.0)*, *fontsize=None*, *example_cellradius=10*, *plot_mode='lineplot'*, *palette={'NCEM': 'C1', 'baseline': 'C0'}*, *color=None*, *save=None*, *suffix='best_by_hyperparam.pdf'*, *show=True*, *return_axs=False*)

Plot best model by hyperparameter.

Parameters

- **graph_model_class** (*str*) – Graph model class.
- **baseline_model_class** (*str*) – Baseline model class.
- **partition_show** (*str*) – Showing partition.
- **metric_show** (*str*) – Showing metric.
- **partition_select** (*str*) – Selected partition.
- **metric_select** (*str*) – Selected metric.
- **param_x** (*str*) – Parameter on x axis.
- **param_hue** (*str*) – Parameter for hue.
- **rename_levels** (*list, optional*) – Rename levels with stated logic.
- **subset_hyperparameters** (*list, optional*) – Subset hyperparameters.
- **cv_mode** (*str*) – Cross validation mode.
- **yaxis_limit** (*tuple, optional*) – y axis limits.
- **xticks** (*list, optional*) – List of x ticks.
- **rotate_xticks** (*bool*) – Whether to rotate x ticks.
- **figsize** (*tuple*) – Figure size.

- **fontsize** (*int, optional*) – Font size.
- **plot_mode** (*str*) – Plotting mode, can be *boxplot*, *lineplot* or *mean_lineplot*.
- **palette** (*dict, optional*) – Palette.
- **color** (*str, optional*) – Color.
- **save** (*str, optional*) – Whether (if not None) and where (path as string given as save) to save plot.
- **suffix** (*str*) – Suffix of file name to save to.
- **show** (*bool*) – Whether to display plot.
- **return_axs** (*bool*) – Whether to return axis objects.
- **example_cellradius** (*Optional[int]*) –

Return type
axis if *return_axs* is True.

ncem.train.GridSearchContainer.plot_target_cell_evaluation

GridSearchContainer.**plot_target_cell_evaluation**(*metric_show*, *metric_select*, *param_x*, *ncols=8*, *show=True*, *save=None*, *suffix='target_cell_evaluation.pdf'*, *return_axs=False*, *yaxis_limit=None*, *panelsize=(3.0, 3.0)*, *sharey=False*)

Plot target cell evaluation.

Parameters

- **metric_show** (*str*) – Showing metric.
- **metric_select** (*str*) – Selected metric.
- **param_x** (*str*) – Parameter on x axis.
- **yaxis_limit** (*tuple, optional*) – y axis limits.
- **panelsize** (*tuple*) – Panel size.
- **ncols** (*int*) – Number of columns.
- **save** (*str, optional*) – Whether (if not None) and where (path as string given as save) to save plot.
- **suffix** (*str*) – Suffix of file name to save to.
- **show** (*bool*) – Whether to display plot.
- **return_axs** (*bool*) – Whether to return axis objects.
- **sharey** (*bool*) –

Return type
axis if *return_axs* is True.

ncem.train.GridSearchContainer.select_cv

GridSearchContainer.**select_cv**(*cv_idx*)

Return key of of cross-validation selected with numeric index.

Parameters
cv_idx (*int*) – Index of cross-validation to plot confusion matrix for.

Return type
cv

Raises
ValueError – *cv_idx* out of scope of cross-validation set

CHAPTER

FOUR

TUTORIALS

We provide tutorials in separate repository.

- A tutorial for fitting and evaluating a interactions model on the MERFISH - brain dataset (interactions).

If you would like to add more tutorials, feel free to contibute or open an issue.

CHAPTER

FIVE

ECOSYSTEM

5.1 squidpy

squidpy provides an environment of tools that can be used to analysis spatial transcriptomnics in python.

5.2 scanpy

scanpy provides an environment of tools that can be used to analysis single-cell data in python.

CHAPTER

SIX

REFERENCE

Command-line interface.

CHAPTER

SEVEN

CONTRIBUTOR GUIDE

Thank you for your interest in improving this project. This project is open-source under the BSD license and highly welcomes contributions in the form of bug reports, feature requests, and pull requests.

Here is a list of important resources for contributors:

- Source Code
- Documentation
- Issue Tracker
- Code of Conduct

7.1 How to add a dataloader

Overview of contributing dataloaders to ncem.

1. **Install ncem.**
   Clone ncem into a local repository from *development* branch and install via pip.

```
cd target_directory
git clone https://github.com/theislab/ncem.git
git checkout development
# git pull  # use this to update your installation
cd ncem  # go into ncem directory
pip install -e .  # install
```

2. **Create a new dataloader in *data.py***
   Your dataloader should be a new class in *data.py* (ideally named by first author, e.g. DataLoaderZhang) and should contain the following functions *_register_celldata*, *_register_img_celldata* and *_register_graph_features*.

   _register_celldata creates an AnnData object called *celldata* of the complete dataset with feature names stored in *celldata.var_names*. Cell type annotations are stored in *celldata.obs*. *celldata.uns['metadata']* should contain the naming conventions of files and columns in obs.

   _register_img_celldata then automatically splits the *celldata* into a dictionary of AnnData object with one AnnData object per image in the dataset.

   _register_graph_features can be added in case of additional graph features, e.g. disease status of images.

   Additionally, each dataloader should have a class attribute *cell_type_merge_dict* which provides a logic on how to merge cell types in any subsequent analysis. This can be helpful when considering datasets with fine cell type annotations and a coarser annotation is wanted.

3. **Make loader public (Optional).**
   You can contribute the data loader to public ncem as code through a pull request. Note that you can also just keep the data loader in your local installation if you do not want to make it public.

```
# make sure you are in the top-level ncem directory from step 1
git add *
git commit  # enter your commit description
# Next make sure you are up to date with dev
git checkout development
git pull
git checkout YOUR_BRANCH_NAME
git merge development
git push  # this starts the pull request.
```

In any case, feel free to open an GitHub issue on on the Issue Tracker.

7.2 How to report a bug

Report bugs on the Issue Tracker.

7.3 How to request a feature

Request features on the Issue Tracker.

7.4 How to set up your development environment

You need Python 3.7+ and the following tools:

- Poetry
- Nox
- nox-poetry

You can install them with:

```
$ pip install poetry nox nox-poetry
```

Install the package with development requirements:

```
$ make install
```

You can now run an interactive Python session, or the command-line interface:

```
$ poetry run python
$ poetry run ncem
```

7.5 How to test the project

Run the full test suite:

```
$ nox
```

List the available Nox sessions:

```
$ nox --list-sessions
```

You can also run a specific Nox session. For example, invoke the unit test suite like this:

```
$ nox --session=tests
```

Unit tests are located in the `tests` directory, and are written using the pytest testing framework.

7.6 How to submit changes

Open a pull request to submit changes to this project against the `development` branch.

Your pull request needs to meet the following guidelines for acceptance:

- The Nox test suite must pass without errors and warnings.
- Include unit tests. This project maintains a high code coverage.
- If your changes add functionality, update the documentation accordingly.

To run linting and code formatting checks before committing your change, you can install pre-commit as a Git hook by running the following command:

```
$ nox --session=pre-commit -- install
```

It is recommended to open an issue before starting work on anything. This will allow a chance to talk it over with the owners and validate your approach.

CHAPTER

EIGHT

CREDITS

8.1 Development Lead

- David Fischer <david.fischer@helmholtz-muenchen.de>
- Anna Schaar <anna.schaar@helmholtz-muenchen.de>

8.2 Contributors

None yet. Why not be the first?

CHAPTER

NINE

CONTRIBUTOR COVENANT CODE OF CONDUCT

9.1 Our Pledge

In the interest of fostering an open and welcoming environment, we as contributors and maintainers pledge to making participation in our project and our community a harassment-free experience for everyone, regardless of age, body size, disability, ethnicity, gender identity and expression, level of experience, nationality, personal appearance, race, religion, or sexual identity and orientation.

9.2 Our Standards

Examples of behavior that contributes to creating a positive environment include:

- Using welcoming and inclusive language
- Being respectful of differing viewpoints and experiences
- Gracefully accepting constructive criticism
- Focusing on what is best for the community
- Showing empathy towards other community members

Examples of unacceptable behavior by participants include:

- The use of sexualized language or imagery and unwelcome sexual attention or advances
- Trolling, insulting/derogatory comments, and personal or political attacks
- Public or private harassment
- Publishing others' private information, such as a physical or electronic address, without explicit permission
- Other conduct which could reasonably be considered inappropriate in a professional setting

9.3 Our Responsibilities

Project maintainers are responsible for clarifying the standards of acceptable behavior and are expected to take appropriate and fair corrective action in response to any instances of unacceptable behavior.

Project maintainers have the right and responsibility to remove, edit, or reject comments, commits, code, wiki edits, issues, and other contributions that are not aligned to this Code of Conduct, or to ban temporarily or permanently any contributor for other behaviors that they deem inappropriate, threatening, offensive, or harmful.

9.4 Scope

This Code of Conduct applies both within project spaces and in public spaces when an individual is representing the project or its community. Examples of representing a project or community include using an official project e-mail address, posting via an official social media account, or acting as an appointed representative at an online or offline event. Representation of a project may be further defined and clarified by project maintainers.

9.5 Enforcement

Instances of abusive, harassing, or otherwise unacceptable behavior may be reported by opening an issue. The project team will review and investigate all complaints, and will respond in a way that it deems appropriate to the circumstances. The project team is obligated to maintain confidentiality with regard to the reporter of an incident. Further details of specific enforcement policies may be posted separately.

Project maintainers who do not follow or enforce the Code of Conduct in good faith may face temporary or permanent repercussions as determined by other members of the project's leadership.

9.6 Attribution

This Code of Conduct is adapted from the Contributor Covenant, version 1.4, available at https://www.contributor-covenant.org/version/1/4/code-of-conduct.html

CHAPTER

TEN

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

n

INDEX

N

O

P

R

S

T

V