TRAPT

Release 0.1.7

Guorui Zhang

Nov 15, 2024

CONTENTS

November 15, 2024

Title TRAPT: A multi-stage fused deep learning framework for transcriptional regulators prediction via integrating large-scale epigenomic data

Maintainer Guorui Zhang <mp798378522@gmail.com>

Description TRAPT is a multi-omics integration framework designed for inferring transcriptional regulator activity from a set of query genes. TRAPT employs a multi-stage fusion strategy to address the issues of incomplete cis-regulatory profile coverage and TRBP problems. By leveraging two-stage self-knowledge distillation to extract the activity embedding of regulatory elements, TRAPT can predicts key regulatory factors for sets of query genes through a fusion strategy.

CHAPTER

ONE

TRAPT.CALCSAMPLERPMATRIX MODULE

TRAPT.CalcSampleRPMatrix.**dhs2gene**(*args*, *sample*)

Calculate the Epi regulatory potential score.

Parameters: args : argparse.Namespace

Global parameters.

sample
[str] Epi sample name.

vec
[np.array] Epi PRE score.

Returns
Epi sample name, and Epi-RP score.

CHAPTER

TWO

TRAPT.CALCTRAUC MODULE

class TRAPT.CalcTRAUC.**CalcTRAUC**(*args*, *RP_Matrix_TR_Sample*, *w*)

Bases: `object`

Calculate the area under the curve (AUC) for each TR curve.

args

Global parameters for TRAPT.

Type
TRAPT.Tools.Args

RP_Matrix_TR_Sample

The sum of TR-RP scores and D-RP scores.

Type
anndata.AnnData

w

U-RP scores.

Type
np.array

Notes

The input is the RP matrix, and the calculation is performed as follows:

$$IRP = (TRRP + DRP) \times URP$$

static get_auc(*params*)

Parallel computing module.

Parameters: i : int

The i-th row of the I-RP matrix.

j
[int] Default is 0.

labels
[np.array] Gene vector.

vec
[np.array] I-RP vector.

Returns
i, j, and the AUC score of the i-th TR.

`iter_params`(*gene_vec*, *trunk*)

Parallel parameter module.

Parameters: gene_vec : np.array

Gene vector.

trunk
[int] Number of blocks.

Returns
An iterator.

`run`()

TR auc calculation module execution entry point.

Returns
A pd.DataFrame of AUC scores for TRs.

CHAPTER

THREE

TRAPT.CALCTRRPMATRIX MODULE

TRAPT.CalcTRRPMatrix.**dhs2gene**(*params*)

Calculate the TR regulatory potential score.

Parameters: args : argparse.Namespace

Global parameters.

sample
[str] TR sample name.

vec
[np.array] TR PRE score.

Returns
TR sample name, and TR-RP score.

TRAPT.CalcTRRPMatrix.**str2bool**(*v*)

CHAPTER

FOUR

TRAPT.CALCTRSAMPLERPMATRIX MODULE

class TRAPT.CalcTRSampleRPMatrix.**CalcTRSampleRPMatrix**(*library='library'*, *output='library'*, *type='H3K27ac'*)

Bases: object

run()

CHAPTER

FIVE

TRAPT.DLFS MODULE

class TRAPT.DLFS.**CustomSigmoid**(**args*, ***kwargs*)

Bases: Layer

call(*x*)

This is where the layer's logic lives.

The *call()* method may not create state (except in its first invocation, wrapping the creation of variables or other resources in *tf.init_scope()*). It is recommended to create state, including *tf.Variable* instances and nested *Layer* instances,

in *__init__()*, or in the *build()* method that is

called automatically before *call()* executes for the first time.

Parameters

- **inputs** – Input tensor, or dict/list/tuple of input tensors. The first positional *inputs* argument is subject to special rules: - *inputs* must be explicitly passed. A layer cannot have zero

  arguments, and *inputs* cannot be provided via the default value of a keyword argument.

  - NumPy array or Python scalar values in *inputs* get cast as tensors.
  - Keras mask metadata is only collected from *inputs*.
  - Layers are built (*build(input_shape)* method) using shape info from *inputs* only.
  - *input_spec* compatibility is only checked against *inputs*.
  - Mixed precision input casting is only applied to *inputs*. If a layer has tensor arguments in **args* or ***kwargs*, their casting behavior in mixed precision should be handled manually.
  - The SavedModel input specification is generated using *inputs* only.
  - Integration with various ecosystem packages like TFMOT, TFLite, TF.js, etc is only supported for *inputs* and not for tensors in positional and keyword arguments.

- ***args** – Additional positional arguments. May contain tensors, although this is not recommended, for the reasons above.
- ****kwargs** – Additional keyword arguments. May contain tensors, although this is not recommended, for the reasons above. The following optional keyword arguments are reserved: - *training*: Boolean scalar tensor of Python boolean indicating

  whether the *call* is meant for training or inference.

- *mask*: Boolean input mask. If the layer's *call()* method takes a *mask* argument, its default value will be set to the mask generated for *inputs* by the previous layer (if *input* did come from a layer that generated a corresponding mask, i.e. if it came from a Keras layer with masking support).

Returns
A tensor or list/tuple of tensors.

class TRAPT.DLFS.**FeatureSelection**(*args*, *data_ad*, *type*)

Bases: object

TSFS(*X*, *T*)

Teacher-Student Feature Selection.

Parameters: X : np.array

Epi-RP matrix.

T
[str] Input genes vector.

Returns
Index values sorted by Epi sample weights, and Epi sample weights.

get_act(*t=1*)

U-RP teacher model activation function.

Parameters: t : float

Temperature value.

get_corr(*v1*, *v2*)

Correlation calculation.

get_loss()

U-RP teacher model loss function.

run()

Method execution entry point.

Returns
A pd.DataFrame of U-RP scores for query Genes, and selected sample information.

sort_by_group(*vec*)

Grouping function.

train(*X*, *y*)

U-RP model training entry function.

Parameters: X : np.array

Epi-RP matrix.

y
[str] Input genes vector.

Returns
U-RP model.

class TRAPT.DLFS.**SparseGroupLasso**(*l1=0.01*, *l2=0.01*, *groups=None*)

Bases: Regularizer

get_config()

Returns the config of the regularizer.

An regularizer config is a Python dictionary (serializable) containing all configuration parameters of the regularizer. The same regularizer can be reinstantiated later (without any saved state) from this configuration.

This method is optional if you are just training and executing models, exporting to and from SavedModels, or using weight checkpoints.

This method is required for Keras *model_to_estimator*, saving and loading models to HDF5 formats, Keras model cloning, some visualization utilities, and exporting models to and from JSON.

Returns
Python dictionary.

TRAPT.DLFS.**seed_tensorflow**(*seed=2023*)

CHAPTER

SIX

TRAPT.DLVGAE MODULE

class TRAPT.DLVGAE.**CVAE**(*input_dim*, *condition_dim*, *h_dim*, *z_dim*)

Bases: Module

forward(*x*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

Note: Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

kl_div()

predict_h(*x*)

reparametrize(*mu*, *logstd*)

Return type
Tensor

training: bool

class TRAPT.DLVGAE.**CalcSTM**(*RP_Matrix*, *type*, *checkpoint_path*, *device='cuda'*)

Bases: object

D-RP model network reconstruction module.

Parameters: RP_Matrix : TRAPT.Tools.RP_Matrix

TR-RP matrix and Epi-RP matrix.

type
[str] Epi-RP type.

checkpoint_path
[str] Model save path.

device
[str, optional] cpu/cuda.

static get_cos_similar_matrix(*m1*, *m2*)

Matrix cosine similarity calculation.

get_edge_index(*A*, *B*, *n=10*)

Construct a heterogeneous network.

Parameters: A : anndata.AnnData

TR-RP matrix.

B

[anndata.AnnData] Epi-RP matrix.

n

[int] Number of nearest neighbors for TR.

Returns

TR-Epi heterogeneous network.

init_cvae()

D-RP teacher model training function.

init_vgae(*h*, *use_kd*)

D-RP student model training function.

Parameters: h : torch.Tensor

Potential representation of the D-RP teacher model.

use_kd

[bool] Utilize knowledge distillation.

recon_loss(*z*, *data*, *norm*, *weight*)

Variational Gaussian Autoencoder (VGAE) reconstruction loss.

run(*use_kd=True*)

save_graph()

static sparse_to_tensor(*data*, *type='sparse'*)

class TRAPT.DLVGAE.**InnerProductDecoderWeight**(*A_e*, **args*, ***kwargs*)

Bases: InnerProductDecoder

forward(*z*, *edge_index=None*, *sigmoid=True*)

Decodes the latent variables z into edge probabilities for the given node-pairs edge_index.

Return type

Tensor

Parameters

- **z** (*torch.Tensor*) – The latent space **Z**.
- **sigmoid** (*bool, optional*) – If set to False, does not apply the logistic sigmoid function to the output. (default: True)

training: bool

class TRAPT.DLVGAE.**VariationalGCNEncoder**(*in_channels*, *h_dim*, *z_dim*)

Bases: Module

forward(*x*, *edge_index*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

Note: Although the recipe for forward pass needs to be defined within this function, one should call the `Module` instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

predict_h(*x*, *edge_index*)

training: bool

TRAPT.DLVGAE.**seed_torch**(*seed=2023*)

CHAPTER

SEVEN

TRAPT.RUN MODULE

TRAPT.Run.**main**()

TRAPT method entry function.

TRAPT.Run.**runTRAPT**(*args*)

TRAPT execution entry function.

TRAPT.Run.**args**

Global parameters for TRAPT.

Type
TRAPT.Tools.Args

TRAPT.Run.**Returns**

A pd.DataFrame of TR activity.

TRAPT.Run.**str2bool**(*v*)

CHAPTER

EIGHT

TRAPT.TOOLS MODULE

class TRAPT.Tools.**Args**(*input*, *output*, *library='library'*, *threads=16*, *trunk_size=32768*, *background_genes=6000*, *use_kd=True*, *tr_type='all'*, *source='all'*)

Bases: `object`

TRAPT Global Parameters.

input

Input path for the gene set.

Type
str

output

Output path for TRAPT results.

Type
str

library

Path to the background library, default is the 'library' path in the current directory.

Type
str, optional

threads

Number of processes used for TRAPT inference.

Type
int, optional

trunk_size

Size of the chunks.

Type
int, optional

background_genes

Number of background genes selected.

Type
str, optional

use_kd

Use knowledge distillation.

Type
str, optional

tr_type

all/tf/tcof/cr.

Type
str, optional

source

all/cistrome/chip_altas/gtrd/remap/chip-atlas/remap/encode/geo.

Type
str, optional

class TRAPT.Tools.**RPMatrix**(*library*, *name*, *to_array=True*)

Bases: object

add(*data*)

binarization()

get_data()

Return type
AnnData

minmax_scale(*axis=1*)

norm(*type='l2'*, *axis=1*)

standard_scale(*axis=1*)

class TRAPT.Tools.**RP_Matrix**(*library*)

Bases: object

class TRAPT.Tools.**Type**

Bases: object

ATAC = 'ATAC'

H3K27ac = 'H3K27ac'

CHAPTER

NINE

MODULE CONTENTS

PYTHON MODULE INDEX

t

INDEX

O

P

R

S

T

U

V

W