pimkl Documentation

Release 0.1.1

Joris Cadow and Matteo Manica

Dec 21, 2022

CONTENTS:

CHAPTER

ONE

PIMKL

pathway induced multiple kernel learning for computational biology

- Free software: MIT license
- Documentation: https://pimkl.readthedocs.io.

1.1 Features

The pimkl command:

```
Usage: pimkl [OPTIONS] NETWORK_CSV_FILE NETWORK_NAME GENE_SETS_GMT_FILE
         GENE_SETS_NAME PREPROCESS_DIR OUTPUT_DIR CLASS_LABEL_FILE [LAM]
         [K] [NUMBER_OF_FOLDS] [MAX_PER_CLASS] [SEED] [MAX_PROCESSES]
         [FOLD]

Console script for a complete pimkl pipeline, including preprocessing and
analysis. For more details consult the following console scripts, which
are here executed in this order. `pimkl-preprocess --help` `pimkl-analyse
run-performance-analysis --help`

Options:
-fd, --data_csv_file PATH       [required]
-nd, --data_name TEXT           [required]
--model_name [EasyMKL|UMKLKNN|AverageMKL]
--help                          Show this message and exit.
```

1.2 Requirements

- C++14 capable C++ compiler
- cmake (>3.0.2)
- Python

1.3 Installation

Install the dependencies

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

Install the package

```
pip install .
```

1.4 Tutorial

You can find a brief tutorial in the dedicated folder.

1.5 Web service deprecation

The PIMKL web-service has been deprecated in favour of the python package hosted in this repository. Please check the examples and the tutorials to use PIMKL in your research.

1.6 Credits

This package was created with Cookiecutter and the audreyr/cookiecutter-pypackage project template.

CHAPTER

TWO

INSTALLATION

2.1 Stable release

To install pimkl, run this command in your terminal:

```
$ pip install pimkl
```

This is the preferred method to install pimkl, 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 pimkl can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/PhosphorylatedRabbits/pimkl
```

Or download the tarball:

```
$ curl -OJL https://github.com/PhosphorylatedRabbits/pimkl/tarball/master
```

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

```
$ pip install .
```

CHAPTER

THREE

PIMKL TUTORIAL

3.1 Data retrieval

You can download the data from the tutorial from here.

In the following we assume you placed files in a folder called data with the following structure:

```
data
├── data.csv
├── gene_sets.gmt
├── interactions.csv
└── labels.csv

0 directories, 4 files
```

Please, check carefully the data format in case you want to run `pimkl` on your data.

3.2 Installation

For the installation of `pimkl` we suggest to follow the description reported here.

3.3 Run `pimkl`

The `pimkl` script reproduces the output that can be obtained from the PIMKL web service.

3.3.1 Pipeline execution

You can run the full `pimkl` pipeline (supervised) by executing:

```
pimkl -fd data/data.csv -nd tutorial --model_name EasyMKL data/interactions.csv network␣
↪data/gene_sets.gmt genes data/preprocess data/output data/labels.csv
```

You can change the parameters (e.g., regularization, number of folds) by providing them to the script:

```
pimkl --help
Usage: pimkl [OPTIONS] NETWORK_CSV_FILE NETWORK_NAME GENE_SETS_GMT_FILE
             GENE_SETS_NAME PREPROCESS_DIR OUTPUT_DIR CLASS_LABEL_FILE [LAM]
```

(continues on next page)

(continued from previous page)

```
              [K] [NUMBER_OF_FOLDS] [MAX_PER_CLASS] [SEED] [MAX_PROCESSES]
              [FOLD]

  Console script for a complete pimkl pipeline, including preprocessing and
  analysis. For more details consult the following console scripts, which
  are here executed in this order. `pimkl-preprocess --help` `pimkl-analyse
  run-performance-analysis --help`

Options:
  -fd, --data_csv_file PATH       [required]
  -nd, --data_name TEXT           [required]
  --model_name [EasyMKL|UMKLKNN|AverageMKL]
  --help                          Show this message and exit.
```

For example:

```
pimkl -fd data/data.csv -nd tutorial --model_name EasyMKL data/interactions.csv network␣
↪data/gene_sets.gmt genes data/preprocess data/output data/labels.csv 0.2 5 50
```

Tip: increasing the number of folds is useful to have better estimates of the significant gene sets/pathways.

3.3.2 Stepwise execution

`pimkl` pipeline can be also run in a stepwise fashion.

Preprocessing

`pimkl-preprocess` prepares the pathway-specific inducers and the data for the subsequent analysis:

```
pimkl-preprocess --help
Usage: pimkl-preprocess [OPTIONS] NETWORK_CSV_FILE NETWORK_NAME
                        GENE_SETS_GMT_FILE GENE_SETS_NAME PREPROCESS_DIR

  Compute incuding Laplacian matrices and preprocess data matrices for
  matching features.

  Multiple data_csv_files may be passed. Each data_csv_file should readable
  as pandas.DataFrames `pd.read_csv(filename, sep=',', index_col=0)` where
  index are features (rows) and columns a are samples.

  The `network_csv_file` is an edge list readable with
  `pd.read_csv(filename)` where the 3rd columns is a numeric value.

  The `gene_sets_gmt_file` should follow the gmt specification. See http://s
  oftware.broadinstitute.org/cancer/software/gsea/wiki/index.php/Data_format
  s

  For each file, a name has to be passed. Names cannot contain "_" or "-".

  Results are written to `preprocess_dir`.
```

(continues on next page)

(continued from previous page)

```
Options:
  -fd, --data_csv_file PATH  [required]
  -nd, --data_name TEXT      [required]
  --help                     Show this message and exit.
```

Execute it on the tutorial data by running:

```
pimkl-preprocess -fd data/data.csv -nd tutorial data/interactions.csv network data/gene_
↪sets.gmt genes data/preprocess
```

Analysis

`pimkl-analyse` is responsible of analysing the preprocessed data.

```
pimkl-analyse --help
Usage: pimkl-analyse [OPTIONS] COMMAND [ARGS]...

Options:
  --help  Show this message and exit.

Commands:
  run-kpca                  KernelPCA with given pathway weights
  run-performance-analysis  train and test many folds
```

Here we focus on the component `pimkl-analyse run-performance-analysis`, to obtain prediction performance and an estimate of the most significant gene sets/pathways:

```
pimkl-analyse run-performance-analysis --help
Usage: pimkl-analyse run-performance-analysis [OPTIONS] NETWORK_NAME
                                              GENE_SETS_NAME PREPROCESS_DIR
                                              OUTPUT_DIR CLASS_LABEL_FILE
                                              [LAM] [K] [NUMBER_OF_FOLDS]
                                              [MAX_PER_CLASS] [SEED]
                                              [MAX_PROCESSES]

  Run classifications using pathway induced multiple kernel learning on
  preprocessed data and inducers on a number of train/test splits and
  analyse the resulting classification performance and learned pathway
  weights.

  The `class_label_file` should be readable with `pd.read_csv(
  class_label_file, index_col=0, header=None, squeeze=True)`

Options:
  -nd, --data_name TEXT           [required]
  --model_name [EasyMKL|UMKLKNN|AverageMKL]
  --help                          Show this message and exit.
```

Run the analysis on the tutorial data by executing:

```
pimkl-analyse run-performance-analysis -nd tutorial --model_name EasyMKL network genes␣
↪data/preprocess data/output data/labels.csv
```

If you want to have more info/examples on how to use `pimkl` feel free to open an issue with the tag tutorial on the repo.

CHAPTER

FOUR

PIMKL API

4.1 pimkl package

4.1.1 Subpackages

pimkl.cli package

Submodules

pimkl.cli.analyse module

pimkl.cli.analyse.**analyse**(*data_names*, *network_name*, *gene_sets_name*, *preprocess_dir*, *output_dir*, *class_label_file*, *model_name='EasyMKL'*, *lam=0.2*, *k=5*, *number_of_folds=2*, *max_per_class=20*, *seed=0*, *max_processes=2*)

pimkl.cli.analyse.**kpca**(*data_names*, *network_name*, *gene_sets_name*, *preprocess_dir*, *output_dir*, *class_label_file*, *weights_csv_file*, *fold*)

pimkl.cli.analyse.**read_preprocessed**(*data_names*, *network_name*, *gene_sets_name*, *preprocess_dir*)

pimkl.cli.cli module

Console script for pimkl.

pimkl.cli.preprocess module

Main module.

pimkl.cli.preprocess.**assert_valid_names**(**names*)

pimkl.cli.preprocess.**invalid_name**(*name*)

pimkl.cli.preprocess.**preprocess_data_and_inducers**(*data_csv_files*, *data_names*, *network_csv_file*, *network_name*, *gene_sets_gmt_file*, *gene_sets_name*, *preprocess_dir*, *match_samples*)

Inducers, that is Laplacian matrices for geneset subnetworks, and data are preprocessed and written to file. Data and inducers are filtered for genes (per dataset) available in the data and the network and the union of genesets. Conditionally, also the data is filtered for matching samples over all datasets.

pimkl.cli.preprocess.**read_data**(*filename*, *reader*, *gene_name_transformation=None*)

pimkl.cli.preprocess.**reader**(*filename*)

Module contents

pimkl.factories package

Submodules

pimkl.factories.estimator_factory module

estimator_factory.**ESTIMATOR_FACTORY = {'EasyMKL': pymimkl.EasyMKL, 'SVC': <class 'sklearn.svm._classes.SVC'>}**

pimkl.factories.induction_factory module

induction_factory.**INDUCTION_FACTORY = {'induce_gaussian_kernel': pymimkl.induce_gaussian_kernel, 'induce_linear_kernel': pymimkl.induce_linear_kernel, 'induce_polynomial_kernel': pymimkl.induce_polynomial_kernel, 'induce_sigmoidal_kernel': pymimkl.induce_sigmoidal_kernel}**

pimkl.factories.mkl_factory module

class pimkl.factories.mkl_factory.**WeightedAverageMKL**(**args: Any*, ***kwargs: Any*)

Bases: AverageMKL

small wrapping of AverageMKL where the additional cunstructor parameter kernels_weights is used to predict a final kernel rather than the average.

The applied weights are corrected to sum up to one.

fit(*X*, *y=None*)

class pimkl.factories.mkl_factory.**WeightedAverageMKL**(**args: Any*, ***kwargs: Any*)

Bases: AverageMKL

small wrapping of AverageMKL where the additional cunstructor parameter kernels_weights is used to predict a final kernel rather than the average.

The applied weights are corrected to sum up to one.

fit(*X*, *y=None*)

mkl_factory.**MKL_FACTORY = {'AverageMKL': pymimkl.AverageMKL, 'EasyMKL': pymimkl.EasyMKL, 'UMKLKNN': pymimkl.UMKLKNN, 'WeightedAverageMKL': <class 'pimkl.factories.mkl_factory.WeightedAverageMKL'>}**

Module contents

pimkl.models package

Submodules

pimkl.models.pimkl module

Pathway Induced Multiple Kernel Learning.

class pimkl.models.pimkl.**PIMKL**(*inducers*, *induction='induce_linear_kernel'*, *mkl='UMKLKNN'*, *estimator='EasyMKL'*, *induction_parameters={}*, *mkl_parameters={'epsilon': 0.0001, 'k': 5, 'kernel_normalization': True, 'maxiter_qp': 100000, 'precompute': True}*, *estimator_parameters={'epsilon': 1e-05, 'kernel_normalization': False, 'lam': 0.2, 'precompute': True, 'regularization_factor': False}*)

Bases: BaseEstimator, ClassifierMixin

Pathway Induced Multiple Kernel Learning with choice of MKL and estimator algorithm. Estimator is only trained when MKL is not an estimator itself.

fit(*X*, *y=None*)

Fit the model. Estimator is only trained when MKL is not an estimator.

get_params(*deep=True*)

Get model parameters.

predict(*X*)

Predict using trained model.

It returns the optimal kernel using learned weights or, in case labels were fitted in training, the predicted labels.

predict_proba(*X*)

Predict probabilities using trained model.

set_estimator_params(***parameters*)

Set model parameters.

set_mkl_params(***parameters*)

Set model parameters.

set_params(***parameters*)

Set model parameters.

Module contents

pimkl.utils package

Subpackages

pimkl.utils.preprocessing package

Submodules

pimkl.utils.preprocessing.core module

Core data pre-processing utilities.

pimkl.utils.preprocessing.core.**enforce_pandas_dataframe_on_second_argument**(*function*)

Decorate to enforce pandas DataFrame argument as input.

pimkl.utils.preprocessing.core.**labels_to_one_hot_code**(*labels*, *n=None*)

Transform labels to one-hot-code.

pimkl.utils.preprocessing.core.**labels_to_one_hot_code_using_dict**(*labels*, *labels_dict*)

Transform labels to one-hot-code.

pimkl.utils.preprocessing.scaler module

Data scaling utilities.

class pimkl.utils.preprocessing.scaler.**Scaler**(*min_target=0.0*, *max_target=1.0*, *fill_value=0.0*)

Bases: object

Object for data scaling.

apply(*second*)

reapply(*second*)

scales = None

unapply(*second*)

pimkl.utils.preprocessing.standardizer module

Data standardization utilities.

class pimkl.utils.preprocessing.standardizer.**Standardizer**

Bases: object

Object for data standardization.

apply(*second*)

parameters = None

reapply(*second*)

unapply(*second*)

Module contents

Data pre-processing utilities.

Submodules

pimkl.utils.objects module

pimkl.utils.objects.**is_sequence**(*arg*)

pimkl.utils.objects.**is_sequence_of_sequence**(*arg*)

Module contents

4.1.2 Submodules

4.1.3 pimkl.analysis module

pimkl.analysis.**plot_aucs_to_buffer**(*df*, *save=False*)

plot AUC for multiindexed pandas.DataFrame where df.columns.names = ['data', 'kind']

pimkl.analysis.**plot_weights_significant_correlations_to_buffer**(*weights_df*, *correlation_type*, *save=False*)

plot heatmap showing value of correlation if significant between different molecular signatures where weights_df.index.names is ['fold', 'class']

pimkl.analysis.**plot_weights_to_buffer**(*weights_df*, *save=False*, *plot_correlations=False*)

plot molecular signature over many folds in pathway wise boxes. For non-binary problems each 1 versus Rest is ploted. weights_df.index.names should be ['fold'] or ['fold', 'class'])

pimkl.analysis.**significant_correlations**(*rho*, *sample_length*)

pimkl.analysis.**significant_pathways**(*df*, *alpha=0.001*)

significance test vs the average weight

4.1.4 pimkl.data module

Split data into training and test.

pimkl.data.**get_learning_data**(*X*, *labels=None*, *max_per_class=30*)

Return splitted test and training data for single data type.

pimkl.data.**get_learning_data_in_dict_mode**(*X*, *labels=None*, *data_types=None*, *max_per_class=30*)

Return splitted test and training data for multiple data types.

pimkl.data.**get_learning_data_in_dict_mode_fraction**(*X*, *labels=None*, *data_types=None*, *fraction=0.5*)

Return splitted test and training data for multiple data types.

pimkl.data.**get_learning_data_indices_fraction**(*X*, *fraction=0.5*)

Return data in dict mode splitted using a fraction.

4.1.5 pimkl.evaluation module

pimkl.evaluation.**performances**(*y_true*, *y_score*)

pimkl.evaluation.**roc_analysis**(*y_test*, *y_score*)

pimkl.evaluation.**roc_multiclass**(*y_test*, *y_score*, *n_classes=None*)

pimkl.evaluation.**roc_two_classes**(*y_test*, *y_score*)

pimkl.evaluation.**sensitivity**(*tp*, *fn*)

pimkl.evaluation.**specificity**(*tn*, *fp*)

4.1.6 pimkl.inducers module

pimkl.inducers.**get_matching_data_and_network**(*data*, *network*)

Interesct data labels with network node labels.

pimkl.inducers.**get_pathway_inducer**(*network*, *gene_set*, *normed=True*)

Get a laplacian based pathway inducer.

pimkl.inducers.**get_pathway_selector**(*network*, *gene_set*)

Get a pathway selector.

pimkl.inducers.**read_gmt**(*gmt_file*)

Read a .gmt file.

pimkl.inducers.**read_gmt_from_file_pointer**(*fp*)

pimkl.inducers.**read_inducer**(*filename*, *size*, *header=None*, *sep=','*)

Read inducer in CSC format.

pimkl.inducers.**write_inducer**(*inducer*, *filename*, *sep=','*)

Write and inducer in COO format.

pimkl.inducers.**write_inducers**(*data*, *network*, *gene_sets*, *data_type*, *network_type*, *output_dir*, *selection_only=False*, *gene_set_type=''*)

Write inducers and data for a specific data-network combination.

pimkl.inducers.**write_preprocessed**(*data*, *data_name*, *network*, *network_name*, *gene_sets*, *gene_sets_name*, *output_dir*)

Write inducers and data for a specific data-network combination.

4.1.7 pimkl.network module

class pimkl.network.**Network**(*graph*, *labels*)

Bases: `object`

get_laplacian(*normed=True*, *return_diag=False*)

get_sub_network(*labels*)

pimkl.network.**filter_interaction_table_by_labels**(*interaction_table*, *labels*)

pimkl.network.**force_undirected_coo_matrix_input**(*row*, *col*, *values*)

pimkl.network.**generate_random_sets**(*number_of_sets*, *max_nodes*, *nodes_labels*, *number_of_nodes=None*)

pimkl.network.**get_fantom5_network**(*fantom5_filename*, ***kwargs*)

pimkl.network.**get_network_from_csv**(*filename*, *sep=','*, ***kwargs*)

pimkl.network.**get_network_from_pandas_interactions_list**(*data*, *adjacency=False*, *threshold=None*, *force_undirected=True*)

pimkl.network.**get_random_scale_free_interaction_df**(*nodes_labels*, *m=5*)

pimkl.network.**get_string_network**(*string_filename*, ***kwargs*)

pimkl.network.**get_unique_rows**(*matrix*, *return_index=True*)

pimkl.network.**is_symmetric**(*m*)

pimkl.network.**scale**(*array*)

pimkl.network.**selected_set_to_weighted_adjacency**(*interaction_table*, *selected_set*, *all_nodes_labels*)

4.1.8 pimkl.pimkl module

Main module.

4.1.9 pimkl.run module

pimkl.run.**fold_generator**(*number_of_folds*, *data*, *labels*, *max_per_class*, *transformer_class=<class 'pimkl.utils.preprocessing.standardizer.Standardizer'>*)

generate class balanced splits of data and labels

pimkl.run.**run_model**(*inducers*, *induction_name*, *mkl_name*, *estimator_name*, *mkl_parameters*, *estimator_parameters*, *induction_parameters*, *inducers_extended_names*, *fold_parameters*)

Run a single fold of the model with data splits from fold_generator.

Arguments are those to PIMKL and then the inducer_names and a dict containing the fold specific arguments. In junction with partial and the fold_generator it can be used for running folds in parallel: `` `list(pool.imap(run_fold, fold_generator(...)))` ``

4.1.10 Module contents

Top-level package for pimkl.

CHAPTER

FIVE

CREDITS

5.1 Development Lead

- Joris Cadow <joriscadow@gmail.com>
- Matteo Manica <drugilsberg@gmail.com>

5.2 Contributors

None yet. Why not be the first?

CHAPTER

SIX

HISTORY

6.1 0.1.1 (2020-11-05)

- Release on PyPI.
- Setup tox tests on travis.
- Setup readthedocs.

6.2 0.1.0 (2019-10-01)

- First release.

CHAPTER

SEVEN

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

p

INDEX

N

P

R

S

U

W