setFTs

Release 0.0.1.0

Ebner Simon

Aug 28, 2022

CONTENTS

CHAPTER

ONE

SETFTS

1.1 setFTs package

1.1.1 Submodules

1.1.2 setFTs.setfunctions module

class setFTs.setfunctions.**SetFunction**

Bases: abc.ABC

A parent class solely for inheritance purposes

gains(*n: int*, *S0*, *maximize=True*)

Helper function for greedy min/max. Finds element that will increase the set function value the most, if added to an input subset.

Parameters

- **n** (*int*) – groundset size
- **S0** (*np.array of type np.int32 or np.bool*) – indicator vector to be improved upon

Returns integer index of element that produces the biggest gain if changed to 1 and the corresponding value gain

Return type (np.array,float)

maximize_greedy(*n: int*, *max_card: int*, *verbose=False*, *force_card=False*)

Greedy maximization algorithm for set functions. (Does not guarantee that the optimal solution is found)

Parameters

- **n** (*int*) – groundset size
- **max_card** (*int*) – upper limit of cardinality up to which the greedy algorithm should check
- **verbose** (*bool*) – flag to enable to print gain information for each cardinality
- **force_card** (*bool*) – flag that forces the algorithm to continue until specified max_card is reached

Returns an np.array indicator vector of booleans that maximizes the setfunction and the evaluated setfunction for that indicator

Return type (np.array,float)

minimize_greedy(*n: int*, *max_card: int*, *verbose=False*, *force_card=False*)
Greedy minimization algorithm for set functions does not guarantee that the optimal solution is found

Parameters

- **n** (*int*) – groundset size
- **max_card** (*int*) – upper limit of cardinality up to which the greedy algorithm should check
- **verbose** (*bool*) – flag to enable to print gain information for each cardinality
- **force_card** (*bool*) – flag that forces the algorithm to continue until specified max_card is reached

Returns an array indicator vector of booleans that minimizes the setfunction and the evaluated setfunction for that indicator

Return type (np.array,float)

class setFTs.setfunctions.**SparseDSFTFunction**(*frequencies: numpy.ndarray[Any, numpy.dtype[numpy.typing._generic_alias.ScalarType]]*, *coefficients: numpy.ndarray[Any, numpy.dtype[numpy.float64]]*, *model: str*, *normalization_flag=False*)
Bases: *setFTs.setfunctions.SetFunction*

export_to_csv(*name='dsft'*)
exports the frequencies and coefficients into a csv file

Parameters **name** (*str*) – name of the newly created file

force_k_sparse(*k*)
creates a k-sparse estimate that only keeps the k largest coefficients

Parameters **k** (*int*) – number of coefficients to keep

Returns a sparseDSFTFunction object with only the k largest coefficients

Return type sparseDSFTFunction

maximize_MIP(*C=1000.0*, *cardinality_constraint=None*)
utilizes a Mixed Integer Program solver to maximize a set function value

Parameters

- **C** (*int*) – parameter for the MIP, if 1000. does not work, try larger values (see https://arxiv.org/pdf/2009.10749.pdf)
- **cardinality_constraint** (*int -> bool*) – function that evaluates to true if the cardinality constraint is met. Takes an integer as an input and evaluates to a bool (e.g cardinality_constraint=lambda x: x == 3)

Returns bitvector with the largest function value and associated function value

Return type (npt.NDArray[bool],float)

minimize_MIP(*C=1000.0*, *cardinality_constraint=None*)
utilizes a Mixed Integer Program solver to minimize a set function value

Parameters

- **C** (*int*) – parameter for the MIP, if 1000. does not work, try larger values (see https://arxiv.org/pdf/2009.10749.pdf)

- **cardinality_constraint** (*int -> bool*) – function that evaluates to true if the cardinality constraint is met. Takes an integer as an input and evaluates to a bool (e.g cardinality_constraint=lambda x: x == 3)

Returns bitvector with the smallest function value and associated function value

Return type (npt.NDArray[bool],float)

shapley_values()
Calculates the Shapley Values for all elements in the ground set

Returns an np.array the length of the groundset of shapley values

Return type npt.NDArray[float64]

spectral_energy(*max_card*, *flag_rescale=True*)
Calculates the spectral energy for each cardinality

Parameters

- **max_card** (*int*) – Maximum Cardinality to consider
- **flag_rescale** – flag indicating whether spectral energy per cardinality should be rescaled to be relative to the total energy

Flag_rescale bool

Returns spectral energy per cardinality

Return type List[float]

class setFTs.setfunctions.**WHTOneHop**(*n*, *weights*, *set_function*, *model*)
Bases: *setFTs.setfunctions.SetFunction*

class setFTs.setfunctions.**WrapSetFunction**(*s: Callable[[numpy.ndarray[Any, numpy.dtype[numpy.typing._generic_alias.ScalarType]]], float]*, *n*, *use_call_dict=False*, *use_loop=True*)
Bases: *setFTs.setfunctions.SetFunction*

Wrapper class for instantiating set functions with a callable function

transform_fast(*model='3'*)
fast Fourier transformation algorithm (not advised)

Parameters **model** (*str*) – basis upon which to calculate the transform see arxiv.org/pdf/2001.10290.pdf for more info

Returns a sparseDSFTFunction object of the desired model

Return type sparseDSFTFunction

transform_sparse(*model='3'*, *k_max=None*, *eps=1e-08*, *flag_print=True*, *flag_general=True*)
sparse Fourier transformation algorithm

Parameters **model** (*str*) – basis upon which to calculate the Fourier transform

Returns a sparseDSFTFunction object of the desired model

Return type sparseDSFTFunction

class setFTs.setfunctions.**WrapSignal**(*signal: List[float]*)
Bases: *setFTs.setfunctions.SetFunction*

Wrapper class for instantiating set functions with a full list of set function evaluations

export_to_csv(*name='sf.csv'*)
exports the frequencies and coefficients into a csv file

Parameters

- **name** – name of the newly created file ending in .csv
- **type** – str

max()
finds the subset that returns the largest set function value

Returns indicator vector that maximizes the set function

Return type npt.NDArray[bool]

min()
finds the subset that returns the smallest set function value

Returns indicator vector that minimizes the set function

Return type npt.NDArray[bool]

spectral_energy(*max_card*, *flag_rescale=True*)
calculates the normalized coefficients per cardinality

Parameters

- **max_card** (*int*) – maximum cardinality for which to calculate the spectral energy
- **flag_rescale** (*int*) – flag indicating whether to average over all coefficients

Returns normalized coefficient of length max_card

Return type List[float]

transform_fast(*model='3'*)
fast Fourier transformation algorithm

Parameters **model** (*str*) – basis upon which to calculate the transform see arxiv.org/pdf/2001.10290.pdf for more info

Returns sparseDSFTFunction object of the desired model

Return type sparseDSFTFunction

transform_sparse(*model='3'*, *k_max=None*, *eps=1e-08*, *flag_print=True*, *flag_general=True*)
sparse Fourier transformation algorithm

Parameters

- **model** (*str*) – basis upon which to calculate the Fourier transform
- **k_max** (*int*) – max number of coefficients to keep track of during computation
- **eps** (*float of form 1e-i*) – eps: abs(x) < eps is treated as zero
- **flag_print** (*bool*) – enables verbose mode for more information
- **flag_general** (*bool*) – enables random one-hop Filtering

Returns a sparseDSFTFunction object of the desired model

Return type sparseDSFTFunction

setFTs.setfunctions.**build_from_csv**(*path*, *model=None*)
loads a setfunction from a csv file

Parameters

- **path** (*str*) – path to csv file

- **model** (*str*) – DSFT model to build (None to build set function)

Returns SparseDSFTFunction or WrapSignal

setFTs.setfunctions.**createRandomSparse**(*n*, *k*, *constructor*, *rand_sets=<function <lambda>>*, *rand_vals=<function <lambda>>*)

creates a random k-sparse set function

Parameters

- **n** – size of the ground set
- **k** – desired sparsity
- **constructor** – a Fourier Sparse SetFunction constructor
- **rand_sets** – a random zero-one vector generator
- **rand_vals** – a random Fourier coefficient generator

Returns a fourier sparse set function, the actual sparsity

setFTs.setfunctions.**eval_sf**(*gt: setFTs.setfunctions.SetFunction*, *estimate: setFTs.setfunctions.SetFunction*, *n: int*, *n_samples=1000*, *err_types=['rel']*, *custom_samples=None*, *p=0.5*)

evaluation function for setfunction. Compares an estimation to the ground truth

Parameters

- **gt** – a SetFunction representing the ground truth
- **estimate** – a SetFunction
- **n** – the size of the ground set
- **n_samples** – number of random measurements for the evaluation
- **err_types** – List of strings that are mae or relative reconstruction error

Returns error values

Return type List[float]

1.1.3 setFTs.plotting module

setFTs.plotting.**plot_freq_card**(*sf*, *plot_type='bar'*)
plot the number of frequencies per cardinality

Parameters

- **sf** (setfunctions.SetFunction) – SetFunction object
- **plot_type** (*str*) – specifies plot type. Either 'bar' or 'plot

setFTs.plotting.**plot_freq_card_multi**(*sf_list*, *label_list*, *plot_type='bar'*)
plot the number of frequencies per cardinality for multiple setfunctions

Parameters

- **sf_list** (*List[setfunctions.SetFunctions]*) – list of SetFunction objects
- **label_list** (*List[str]*) – list of labels for the setfunctions in corresponding order
- **plot_type** (*str*) – specifies plot type. Either 'bar' or 'plot

setFTs.plotting.**plot_max_greedy**(*sf_list*, *label_list*, *n*, *max_card*)
plots the result of the greedy maximization when restricted to each cardinality

Parameters

- **sf_list** (*List[setfunctions.SetFunctions]*) – list of SetFunction objects
- **label_list** (*List[str]*) – list of labels for the setfunctions in corresponding order
- **n** (*int*) – ground set size
- **max_card** (*int*) – maximal cardinality to consider

setFTs.plotting.**plot_max_mip**(*ft_list*, *label_list*, *max_card*)
plots the result of the MIP-based maximization when restricted to each cardinality

Parameters

- **ft_list** (*List[setfunctions.SetFunctions]*) – list of SetFunction objects
- **label_list** (*List[str]*) – list of labels for the setfunctions in corresponding order
- **max_card** (*int*) – maximal cardinality to consider

setFTs.plotting.**plot_min_greedy**(*sf_list*, *label_list*, *n*, *max_card*)
plots the result of the greedy minimization when restricted to each cardinality

Parameters

- **sf_list** (*List[setfunctions.SetFunctions]*) – list of SetFunction objects
- **label_list** (*List[str]*) – list of labels for the setfunctions in corresponding order
- **n** (*int*) – ground set size
- **max_card** (*int*) – maximal cardinality to consider

setFTs.plotting.**plot_min_mip**(*ft_list*, *label_list*, *max_card*)
plots the result of the MIP-based minimization when restricted to each cardinality

Parameters

- **ft_list** (*List[setfunctions.SetFunctions]*) – list of SetFunction objects
- **label_list** (*List[str]*) – list of labels for the setfunctions in corresponding order
- **max_card** (*int*) – maximal cardinality to consider

setFTs.plotting.**plot_minimization_found**(*sf*, *model='3'*, *greedy=False*)
plots the minimal value found when performing a minimization algorithm on an eps sparse approximation

Parameters

- **sf** (setfunctions.SetFunction) – SetFunction object
- **model** (*int*) – Fourier transformation base to consider
- **greedy** (*bool*) – flag indicating whether greedy (True) or MIP based algorithm (False) should be used

setFTs.plotting.**plot_minimization_found_biggest_coefs**(*sf*, *max_sparsity*, *interval*, *model='3'*, *greedy=False*)
plots the minimal value found when performing a minimization algorithm constrained to its biggest coefficients

Parameters

- **sf** (setfunctions.SetFunction) – SetFunction object
- **max_sparsity** (*int*) – maximal sparsity to consider
- **interval** (*int :param model: Fourier transformation base to consider*) – increment of sparsity
- **greedy** (*bool*) – flag indicating whether greedy (True) or MIP based algorithm (False) should be used

setFTs.plotting.**plot_reconstruction_error**(*sf*, *n*, *err_types=['rel']*, *model='3'*, *flag_general=True*)
plots the reconstruction error when approximated with the sparse algorithm with different eps values

Parameters

- **sf** (setfunctions.SetFunction) – SetFunction object
- **err_types** (*List[str]*) – list of error calculations to perform
- **model** (*int*) – Fourier transformation base to consider
- **flag_general** (*bool*) – enables random one hop filtering

setFTs.plotting.**plot_reconstruction_error_biggest_coefs**(*sf*, *n*, *max_sparsity*, *interval*, *err_types=['rel']*, *model='3'*)
plots the reconstruction error when approximated with the sparse algorithm constrained to only the biggest coefs

Parameters

- **sf** (setfunctions.SetFunction) – SetFunction object
- **n** (*int*) – ground set size
- **max_sparsity** (*int*) – maximal sparsity to consider
- **interval** (*int*) – increment of sparsity
- **err_types** (*List[str]*) – list of error calculations to perform
- **model** (*int*) – Fourier transformation base to consider

setFTs.plotting.**plot_scatter**(*sf*, *label*, *max_card*)
plots the coefficients of a setfunction per cardinality as a scatterplot

Parameters

- **sf** (setfunctions.SetFunction) – SetFunction object
- **label** (*str*) – name of the setfunction

- **max_card** (*int*) – maximal cardinality to consider

setFTs.plotting.**plot_spectral_energy**(*sf*, *max_card*, *flag_rescale=True*, *plot_type='plot'*)

plot the average coefficient for each cardinality

Parameters

- **sf** (setfunctions.SetFunction) – SetFunction object
- **max_card** (*int*) – maximal cardinality to consider
- **flag_rescale** (*bool*) – flag that enables normalization
- **plot_type** (*str*) – specifies plot type. Either 'bar' or 'plot

setFTs.plotting.**plot_spectral_energy_multi**(*sf_list*, *label_list*, *max_card*, *flag_rescale=True*, *plot_type='plot'*)

plot the average coefficient for each cardinality for multiple set functions

Parameters

- **sf_list** (*List[setfunctions.SetFunctions]*) – list of SetFunction objects
- **label_list** (*List[str]*) – list of labels for the setfunctions in corresponding order
- **max_card** (*int*) – maximal cardinality to consider
- **flag_rescale** (*bool*) – flag that enables normalization
- **plot_type** (*str*) – specifies plot type. Either 'bar' or 'plot

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

S

INDEX

W