COSgen Documentation

Release

Aymanns, Florian

Jul 18, 2017

Contents:

CHAPTER 1

COSgen

Overview

Contrast optimized stimulus generator (COSgen) is a highly parameterized genetic algorithm implementation to produce optimized stimulus sequences for clinical and preclinical fMRI. Our package is fully compatible with a pure Micro/Python stimulus delivery solution (COSplay), and provides a convenient and well documented API. Because of its modular structure, the implementation is highly adaptable for specific use cases.

Features

- High adaptability for specific use cases
- Full compatability with COSgen
- Custom model specification (design matrix construction, covariance matrix computation)
- Custom fitness measure specification

Dependencies

- Python 2.7 or Python 3.5 and newer
- Numpy
- Scipy
- argh
- nibabel (only for *Make test data* and *Estimation of auto-correlation*)
- matplotlib (only for `cosgen.models.plot_design_matrix` and the example *Optimal block design for contrast detection*)

Installation

The module can be installed using setuptools. Run `python setup.py install` inside the COSgen folder you downloaded. After installation you can use `COSgen`, `make_test_data` and `estimate_autocorr` directly as commands.

References

- COSplay: https://readthedocs.org/projects/cosplay/
- argh: https://pypi.python.org/pypi/argh
- Python: http://www.python.org/
- Setuptools: https://setuptools.readthedocs.io/en/latest/

CHAPTER 2

Examples

Optimal block design for contrast detection

`block_design_for_destection.py`

Custom model

`custom_model.py`

CHAPTER 3

Command line interface

```
usage: COSgen [-h] [-p POPULATION_SIZE] [-l LIBRARY_SIZE]
              [--storage-path STORAGE_PATH] [--seqlength SEQLENGTH]
              [--nstimtypes NSTIMTYPES] [-g GENERATIONS]
              [--survivors SURVIVORS] [--nimmigrants NIMMIGRANTS]
              [--hrflength HRFLENGTH] [-T TR] [-m MODEL_TYPE]

optional arguments:
  -h, --help            show this help message and exit
  -p POPULATION_SIZE, --population-size POPULATION_SIZE
                        20
  -l LIBRARY_SIZE, --library-size LIBRARY_SIZE
                        20
  --storage-path STORAGE_PATH
                        '~/.cosgen'
  --seqlength SEQLENGTH
                        100
  --nstimtypes NSTIMTYPES
                        1
  -g GENERATIONS, --generations GENERATIONS
                        10000
  --survivors SURVIVORS
                        5
  --nimmigrants NIMMIGRANTS
                        4
  --hrflength HRFLENGTH
                        30
  -T TR, --TR TR        1
  -m MODEL_TYPE, --model-type MODEL_TYPE
                        'detection'
```

CHAPTER 4

cosgen package

Submodules

cosgen.algorithms module

This file contains the genetic algorithm. Any extra algorithms should be added in this file.

exception cosgen.algorithms.**MissingFunction**
Bases: exceptions.Exception

Error raised if the function crate passed to 'ga' as 'functions' does not contain all necessary functions for the execution of the genetic algorithm.

cosgen.algorithms.**ga**(*population*, *functions*, *generations*, *nsurvive*, *nimmigrants*, *stat*)
Run genetic algorithm.

This function runs a genetic algorithm on a population with the given arguments.

Parameters

- **population** (*list of cosgen.sequence.Sequence objects*) – Initial population.
- **functions** (*cosgen.function_crate.FunctionCrate object*) – This object has to have at least a mutate and cross_over function as well as one fitness measure.
- **generations** (*int*) – Number of generations(iterations) of the genetic algorithm.
- **nsurvive** (*int*) – Number of survivors after each generation.
- **nimmigrants** (*int*) – Number of immigrants in each generation.
- **stat** (*cosgen.statistics.Statistics object*) – Logs properties of population over generations.

Returns **population** – Population after genetic algorithm.

Return type list of cosgen.sequence.Sequence objects

cosgen.cli module

cosgen.cli.**cli_algorithm**(*population_size=20*, *library_size=20*, *storage_path='~/.cosgen'*, *seqlength=100*, *nstimtypes=1*, *generations=10000*, *survivors=5*, *nimmigrants=4*, *hrflength=30*, *TR=1*, *model_type='detection'*)

cosgen.cli.**main**()

cosgen.cross_over module

This file contains the cross_over function. Any extra cross over functions should be added in this file.

cosgen.cross_over.**cross_over**(*sequence1*, *sequence2*)

Create offspring of sequence1 and sequence2.

This function creats and offspring of sequence1 and sequence2 by cutting them at a random point and merging the two ends.

Parameters

- **sequence1** (cosgen.sequence.Sequence) – Parent one.
- **sequence2** (cosgen.sequence.Sequence) – Parent two.

Returns Offspring for the two sequences given.

Return type *cosgen.sequence.Sequence*

cosgen.fitness_measures module

This file contains fitness measure functions.

exception cosgen.fitness_measures.**OptimalityError**

Bases: exceptions.Exception

Error raised when a optimality is not 'a' or 'd'.

cosgen.fitness_measures.**estimator_variance**(*sequence*, *model*, *optimality*, *contrast=None*)

The optimality of the estimator variances.

This function calculates the a- or d-optimality value of the covariance matrix of the estimators.

Parameters

- **sequence** (cosgen.sequence.Sequence) – Sequence for which the covariance matrix is calculated.
- **model** (cosgen.models.Model) – Model class providing functions for the construction of the design matrix and covariance matrix.
- **optimality** (*string*) – Can be 'a' for a-optimality (trace) or 'd' for d-optimality (determinat).
- **contrast** (*numpy matrix*) – Matrix containing contrast vectors as rows.

Returns optimality value

Return type float

cosgen.function_crate module

class cosgen.function_crate.**FunctionCrate**

add_fitness_measure(*name*, *function*)
Add fitness measure function to object.

This method adds a fitness measure function to the object, that is used in the evaluate fitness method. The function must take a sequences as parameter and retrun a float.

Parameters

- **name** (*string*) – Name of the fitness measure.
- **function** (*function*) – Fitness measure function.

del_cross_over()
Delete cross over function.

This methods deletes the function added using the set_cross_over method.

del_generate_immigrants()
Delete generate immigrants function.

This methods deletes the function added using the set_generate_immigrants method.

del_mutate()
Delete mutate function.

This methods deletes the function added using the set_mutate method.

evaluate_fitness(*sequence*)
Calculate overall fitness measure.

This method calculates the sum of the return values of all fitness measure functions added to the instance of the class.

Parameters **sequence** (cosgen.sequence.Sequence) – Sequence for which the fitness is calculated.

Returns Overall fitness.

Return type float

static find_best(*population*, *n*)
Find n best sequences in population.

This method finds the n sequences with the highest fitness in population.

Parameters

- **population** (*list of cosgen.sequence.Sequence*) – Population of sequences
- **n** (*int*) – Number of sequences returned.

Returns List of best sequences.

Return type list of cosgen.sequence.Sequence

remove_fitness_measure(*name*)
Remove fitness measure.

This method removes a fitness measure previously added with the add_fitness_measure method.

Parameters **name** (*string*) – Name of the fitness measure to be removed.

set_cross_over (*function*)
Set a cross over function.

This methode sets a cross over function used by the genetic algorithm. The function should normally take two cosgen.sequence.Sequence objects as input and return a cosgen.sequence.Sequence object.

Parameters **function** (*function*) – Cross over function.

set_generate_immigrants (*function*)
Set a generate immigrants function.

This methode sets a generate immigrants function used by the genetic algorithm. The function should normally return a list of cosgen.sequence.Sequence objects. If the function as an argument 'cross_over_fct' a cross over function has to be set in advance using the set_cross_over method. The 'cross_over_fct' parameter is then fixed to this function.

Parameters **function** (*function*) – Generate immigrants function.

set_mutate (*function*)
Set a mutate function.

This methode sets a mutate function used by the genetic algorithm. The function should normally take a cosgen.sequence.Sequence object as input and return a cosgen.sequence.Sequence object.

Parameters **function** (*function*) – Mutate function.

exception cosgen.function_crate.**MissingAttrError**
Bases: exceptions.Exception

This error is raised if a FunctionCrates object misses an attribute to complete the requested operation.

exception cosgen.function_crate.**OverwriteAttrError**
Bases: exceptions.Exception

This error is raised if an attribut of a FunctionCrate object already exist.

exception cosgen.function_crate.**RmAttrError**
Bases: exceptions.Exception

This error is raised if an attribut of a FunctionCrate object can not be removed because it does not exist.

exception cosgen.function_crate.**WrongOrderError**
Bases: exceptions.Exception

This error is raised if the functions are added to a FunctionCrate object in the wrong order.

cosgen.function_crate.**partition** (*population*, *left*, *right*, *pivotIndex*)
Helper function for quickselect. (Code from https://rosettacode.org/wiki/Quickselect_algorithm#Python)

cosgen.function_crate.**quickselect** (*population*, *left*, *right*, *k*)
Returns the k-th smallest, (k >= 0), element of population within population[left:right+1] inclusive. Implementation of the quickselect algorithm from https://rosettacode.org/wiki/Quickselect_algorithm#Python.

cosgen.immigrants module

This file contains the function neccessary to generate immigrants. Any alternative functions should be added here.

cosgen.immigrants.**generate_immigrants** (*nimmigrants*, *seqlen*, *nstimtypes*, *block_size*, *cross_over_fct*)
Generate immigrants.

This function generates 'nimmigrants' sequences partially consisting of a block and a random sequence.

Parameters

- **nimmigrants** (*int*) – Number of sequences to be generated.
- **seqlen** (*int*) – Length of the sequences to be generated.
- **nstimtypes** (*int*) – Number of stimulus types of the sequences.
- **block_size** (*int*) – Size of the blocks in the block sequence part. Has to be a divisor of the sequence length.
- **cross_over_fct** (*function*) – Function taking two sequences as parameters and returning one. (e.g. cosgen.cross_over.cross_over)

Returns List of sequence according to parameters.

Return type list of cosgen.sequence.Sequence

cosgen.models module

class cosgen.models.**DetectionModel**(*hrf*, *whitening_mat=None*, *err_cov_mat=None*, *filter-func=<function <lambda>>*, *extra_evs=None*)

Bases: *cosgen.models.Model*

cov_beta(*X*)

Calculate covariance of estimators (betas).

This method calculated the covariance matrix of the estimators for a given design matrix. It employs pre-whitening.

Parameters **X** (*numpy matrix*) – Design matrix.

Returns Covariance matrix of beta.

Return type numpy matrix

design_matrix(*sequence*)

Calculate design matrix.

This method calculates the desing matrix for a given sequence. Colums of the desing matrix are a constant (ones) a linear time course and the convolution of the hrf with the sequence.

Parameters **sequence** (cosgen.sequence.Sequence) – Sequence for which the design matrix is calculated.

Returns Design matrix.

Return type numpy matrix

class cosgen.models.**EstimationModel**(*basis_set*, *whitening_mat=None*, *err_cov_mat=None*, *filter-func=<function <lambda>>*, *extra_evs=None*)

Bases: *cosgen.models.Model*

This class implements a model for estimating the hrf.

The model employes pre-whitening to account for autocorrelation for the errors. Either 'whitening_mat or 'err_cov_mat' must be given.

Parameters

- **basis_set** (*numpy array*) – Array with hrf basis vetors as rows.
- **whitening_mat** (*numpy matrix, optional*) – Whitening matrix.

- **err_cov_mat** (*numpy matrix, optional*) – Error covariance matrix.
- **filterfunc** (*function*) – Filter function takes numpy array as input and returns filtered numpy array (c.f. *gaussian_highpass()*)
- **extra_evs** (*array-like object*) – Extra explenatory variables in form of a 2D array-like object with regressors as collumns. Shapes is (number of extra evs, sequence length).

cov_beta(*X*)
Calculate covariance of estimators (betas).

This method calculated the covariance matrix of the estimators for a given design matrix. It employs pre-whitening.

Parameters **X** (*numpy matrix*) – Design matrix.

Returns Covariance matrix of beta.

Return type numpy matrix

design_matrix(*sequence*)
Calculate design matrix.

This method calculates the desing matrix for a given sequence. Colums of the desing matrix are a constant (ones) a linear time course and the convolution of the basis vetors with the sequence.

Parameters **sequence** (cosgen.sequence.Sequence) – Sequence for which the design matrix is calculated.

Returns Design matrix.

Return type numpy matrix

class cosgen.models.**Model**(*design_matrix_func*, *cov_beta_func*)

cov_beta(*X*)
Retrun covarinace matrix for a given design matrix.

This method execute the 'cov_beta_func' given in the initialisation of the object. The parameter types and return types depend on the particular function.

Parameters **matrix** (*design*) – Design matrix for which the covariance matrix is to be calculated.

Returns Covarnace matrix for the given design matrix.

Return type covarince matrix

design_matrix(*sequence*)
Retrun design matrix for a given sequence.

This method execute the 'design_matrix_func' given in the initialisation of the object. The parameter types and return types depend on the particular function.

Parameters **sequence** – Sequence for which the design matrix is to be calculated.

Returns Design matrix for the given sequence.

Return type design matrix

cosgen.models.**gaussian_highpass**(*data*, *sigma=225*)

cosgen.models.**get_FIR_basis_set**(*length*)

cosgen.models.**get_ICA_basis_set**(*TR*, *length*, *order*)

cosgen.models.**get_ar1_cov**(*dim*, *phi*)

cosgen.models.**get_autocorr_whitening_mat**(*acf*)

cosgen.models.**get_bspline_basis_set**(*TR*, *length*, *order*)

cosgen.models.**get_canonical_basis_set**(*TR*, *length*, *order*)

cosgen.models.**get_fourier_basis_set**(*TR*, *length*, *order*)

cosgen.models.**get_gamma_basis_set**(*TR*, *length*, *order*, *a1*, *b1*, *a2*, *b2*, *c*)

cosgen.models.**get_gamma_hrf**(*TR*, *length*, *a1*, *b1*, *a2*, *b2*, *c*)

cosgen.models.**orthogonalize**(*A*, *v*)
A must contain already orthogonalized vector!!

cosgen.models.**plot_design_matrix**(*mat*)

cosgen.mutate module

This file holds functions to mutate sequences.

exception cosgen.mutate.**InvalidFractionError**
Bases: exceptions.Exception

cosgen.mutate.**mutate**(*sequence*, *mutation_fraction*)
Mutate sequence with given probability.

This function randomly changes 'mutation_fraction' of the entires of the sequence given and returns the changes sequence.

Parameters

- **sequence** (cosgen.sequence.Sequence) – Sequence to be mutated.
- **mutation_fraction** (*float*) – Fraction of sequence elements to be changed. Has to be between 0 and 1.

Returns Altered sequence.

Return type *cosgen.sequence.Sequence*

cosgen.sequence module

exception cosgen.sequence.**BlockSizeError**
Bases: exceptions.Exception

class cosgen.sequence.**Sequence**(*seqlen=None*, *nstimtypes=1*, *seqtype='random'*, *l=None*, *block_size=None*)

dump(*path*, *index=0*, *TR=1*)

get_block_representation()

cosgen.sequence.**estimate_optimal_block_size**(*seqlen*, *fc*)

cosgen.statistics module

This file holds everything related to logging the statistics of the population during execution of the genetic algorithm.

class cosgen.statistics.**Statistics**(*storage_path*)

add(*population*)
Add population statistics to log.

This function stores the maximum fitness, average fitness, and the population diversity (average hamming distance).

Parameters **population** (*list of cosgen.sequence.Sequence*) –

gen_plot()
Generates plot and save it. Add description of what happens if matplotlib is not there

Module contents

CHAPTER 5

Scripts

Make test data

`make_test_data`

Estimation of auto-correlation

`estimate_autocorr`

CHAPTER 6

Indices and tables

- genindex
- modindex
- search

Python Module Index

C

Index

M

O

P

Q

R

S

W