Module **axelrod_model_improvement_groupdependentat**

Simulations based on the Axelrod model and extensions to inspect the volatility of the features over time (AXELROD MODEL + Agreement threshold + two model variations based on the Social identity approach) The Axelrod model is used to predict the number of changes per feature in comparison to the datasets and is used to compare different model variations and their performace.

Input: Real data

–take the number of change per step and not the distance fucntion between the feature vectors as a break criteria vector-version of axelrod –

Description: The Axelrod model depicts convergence and diversity on a macro level, driven by local agent-based interaction mechanisms. The agreement-threshold model (MacCarron et al. 2020a), an extension of the Axel- rod model, acts as a multi-dimensional opinion dynamics model. We extend these agent-based models by explicit aspects of the social identity approach to recover real-world dynamics better and to assess the prediction performance of data simulations. We newly introduce mechanisms on in- and out-group interaction.

Model variations: Parameter: Group-dependent preference - Interaction within a group takes place without the limitation of the agreement threshold - Interaction between groups involves an agreement threshold - Group parameter needed

Parameter: In-group preference - Integrates an interaction preference towards in-group members - Reduced interaction probability with an out-group members - Inter-group interaction still possible, but unlikely - Group parameter needed

author: alejandro dinkelberg date: 12.01.2022

Functions

def count_cluster(f)

count the number of clusters (agents with identical vectors) #no used#

Args

f : `np.array`
cultural vectors of the agents

Returns

`list`
: mean cluster size, number of clusters

`def feature_distribution(n_features=8, features=None, group_array=None)`

get feature distribution for each group

Args

`n_features` : `int` , optional
: *description*. Defaults to 8.

`features` : `_type_` , optional
: *description*. Defaults to None.

`group_array` : `_type_` , optional
: *description*. Defaults to None.

Raises

`ValueError`
: ("group and features do not have the same length")

Returns

`_type_`
: (dict) with attitude distributions

`def number_of_changes(dataset, precise_difference=True)`

define number of changes and give back a dictionary

Args

`dataset` : `string`
: filename

`precise_difference` : `bool` , optional
: precise difference (forth and back are two changes) Defaults to True.

Returns

`_type_`
: (dict) differences, (dict) used data, ID_party_aff, diff_array

```
def run_main(selected_dataset='BJSP', at_value=[1], topo_value=[0], rounds=2000,
             only_same_group_interaction=False, group_dependent_at=False)
```

main function

Args

selected_dataset : `str`, optional
determine dataset. Defaults to 'BJSP'.

at_value : `list`, optional
agreement threshold. Defaults to [1].

topo_value : `list`, optional
topology of the model. Defaults to [0].

rounds : `int`, optional
max number of simulation runs. Defaults to 2000.

only_same_group_interaction : `bool`, optional
model variation 1. Defaults to False.

group_dependent_at : `bool`, optional
model variation 2. Defaults to False.

```
def run_model(grid, n_features, t_max, objective_changes_list, topo, at, connection,
              timepoint_to_measure, data_array, group_array, only_same_group_interaction,
              group_dependent_at, q)
```

callable run function for multiprocessing

Args

grid : `int`
size of grid

n_features : `int`
number of features

t_max : `int`
max running time

objective_changes_list : `list`
maximum number of changes which are simulated by the mode

topo : `int`
underlying topology of the model

at : int
agreement threshold; 0 = no agreement threshold

connection : int
number of links between agents (for example to generate AB network)

timepoint_to_measure : int
How often do we measure the state of the model

data_array : np.array
position of each agent from the data

group_array : np.array
group identifies for every agent

only_same_group_interaction : bool
model variation; interaction highly limited to group members

group_dependent_at : bool
model variation; at for out-group members and no at for in-group members

q : queue
multiprocessing, get back the results (return statement)

Classes

```
class AxelrodModelNumpy (Grid_x=10, Grid_y=10, n_features=8, t_max=100,
                         objective_changes_list=[1000], topo=0, at=0, m_AB=2, memory=False,
                         timepoint_to_measure=10000, data_array=None, group_array=None,
                         only_same_group_interaction=False, group_dependent_at=False)
```

Methods

```
def run_model(self, t_max)
```

Run method of the Axelrod model, is dependent on the number of changes from the data and will stop when the maximum number of possible changes is reached. We only set a limit to secure that it terminates but usually it should not be reached. The model runs and gives the results for :param t_max: number of events :return: set of measurements –> list(number of changes per feature, length of run, number of overall changes, biggest cluster size)

Index

Functions

Classes