Competency-Gain-Prediction

Competency-Gain-Prediction provides packages and analysis tools for Longitudinal Item Response Theory models. The focus of this repository is to estimate and predict the competency gain that is achieved during a learning intervention.

File structure

The file structure contains three packages. The em_algorithm package provides tools for parameter estimation, the models package includes the implemented MIRT models, and the simulation_framework package includes classes that are capable of simulating item response data for the pre-test post-test setting of a learning intervention. The packages include tests that are, among other files, not documented in the file structure below.

```
├── CSEDM Challenge               #Code related to the 2nd CSEDM data challenge
│   ├── explore_data_csedm.ipynb
│   ├── item_skill_relation.csv   #Item Competency relations for CSEDM data
│   ├── knowledge_growth_model_uirt_lfa.py
│   ├── ProgSnap2.py
│   ├── uirt_lfa_model_csedm_final.ipynb
├── em_algorithm                  #EM algorithm package
│   ├── em_algorithm.py
│   ├── e_step.py
│   ├── e_step_mirt_2pl.py        #Inherits from e_step
│   ├── e_step_mirt_2pl_gain.py
│   ├── m_step.py
│   ├── m_step_mirt_2pl.py        #Inherits from m_step
│   ├── m_step_mirt_2pl_gain.py
├── models                        #Models package
│   ├── irt_model.py
│   ├── mirt_2pl.py               #Inherits from irt_model
│   ├── mirt_2pl_gain.py          #Inherits from mirt_2pl
├── simulation_framework
├── experiment_worker.py
├── simulation_experiment.py
├── tables_and_graphics.ipynb
├── requirements.txt
└── README.md
```

Requirements

The following package versions were confirmed to function with the source code of the repository.

```
python=3.8.13
numpy=1.23.4
pandas=1.5.1
matplotlib=3.5.3
scipy=1.9.1
scikit-learn=1.1.3
seaborn=0.12.0
hotelling=0.5.0
cma=3.2.2
girth=0.8.0
```

Usage

There is not yet an easy to use API for obtaining the parameters of a single model. Nevertheless, the repository is focused on simulation experiments with different baselines and multiple repetitions. To test the installation and to conduct such an experiment, the following code snippet can be used.

```
import sys
import os
sys.path.append(os.path.realpath("./models"))
if True:  # noqa: E402
    import simulation_experiment

result_dict = simulation_experiment.mirt_simulation_experiment(
                sample_size=30, item_dimension=10, latent_dimension=2,
                q_type="full",
                early_person_method="BFGS", late_person_method="BFGS",
                methods=["real_early", "pure_competency", "initial",
                            "late_em", "difference", "real_parameters"],
                gain_mean=1.5)

print("Late EM, Initial Competency Covariance:")
print(result_dict["late_em"]["estimated_early_parameters"]["person_parameters"]
["covariance"])
print("Late EM, Full Covariance:")
print(result_dict["late_em"]["estimated_late_parameters"]["person_parameters"]
["covariance"])
```

The function mirt_simulation_experiment() returns an extensive dictionary with all simulation data. For instance, the estimated late covariances of the Late EM Method are printed.

```
Late EM, Initial Competency Covariance:
[[1.     0.6607]
 [0.6607 1.    ]]
Late EM, Full Covariance:
[[ 1.      0.6606 -0.1001 -0.1381]
 [ 0.6606  1.     -0.1183 -0.1214]
 [-0.1001 -0.1183  0.5563  0.4743]
 [-0.1381 -0.1214  0.4743  0.5789]]
```