IPD Evolutionary Training

Group Project for COMP 3710

By: Mio Tanaka, Suraiya Khanda, Samar Houssami, Ben Davidson

This project uses Jupyter Notebooks, Python 3.7, the Alexrod-Python and the Axelrod-Dojo library to run, analyize and visualise an Iterated Prisioners Dilemma Tournament and introduce machine learning strategies with finite state machines.

We made some minor modifications to the dojo library to improve the reporting and output. This is most reflected in the `training_output.csv` which now records detailed information about the players used in the simulation, mutation rate, bottleneck, size of state machine and the date/time to aid in reproducing the results.

We also made minor modifications to the main Axelrod library (`player.py`) to keep strategy name short so they would display correctly in charts and graphs

```
# import IPython
# from IPython.core.display import display, HTML
# display(HTML("<style>.container { width:100% !important; }</style>"))
```

If you are doing serious testing uncomment this block and widen your browser to see the full output and retain clean line breaks

Import the axelrod library

```
import axelrod as axl
%matplotlib inline

from datetime import datetime
print("Run at: " + datetime.now().strftime('%Y-%m-%d %H:%M:%S'))
```

```
Run at: 2019-04-10 19:24:48
```

The parameters that we are working with

Finite State Machine Evolver

```
Usage:
    fsm_evolve.py [-h] [--generations GENERATIONS] [--population POPULATION]
```

```
    [--mu MUTATION_RATE] [--bottleneck BOTTLENECK] [--processes PROCESSORS]
    [--output OUTPUT_FILE] [--objective OBJECTIVE] [--repetitions REPETITIONS]
    [--turns TURNS] [--noise NOISE] [--nmoran NMORAN]
    [--states NUM_STATES]

Options:
    -h --help                   Show help
    --generations GENERATIONS   Generations to run the EA [default: 500]
    --population POPULATION     Population size  [default: 40]
    --mu MUTATION_RATE          Mutation rate [default: 0.1]
    --bottleneck BOTTLENECK     Number of individuals to keep from each generation [default: 10]
    --processes PROCESSES       Number of processes to use [default: 1]
    --output OUTPUT_FILE        File to write data to [default: fsm_params.csv]
    --objective OBJECTIVE       Objective function [default: score]
    --repetitions REPETITIONS   Repetitions in objective [default: 100]
    --turns TURNS               Turns in each match [default: 200]
    --noise NOISE               Match noise [default: 0.00]
    --nmoran NMORAN             Moran Population Size, if Moran objective [default: 4]
    --states NUM_STATES         Number of FSM states [default: 8]
```

Import dojo

```
import axelrod_dojo as dojo
objective = dojo.prepare_objective(name="score", turns=10, repetitions=1)

params_class = dojo.FSMParams
# params_class = dojo.HMMParams
params_kwargs = {"num_states": 2}
```

In this example we use a small number of states (2). This allows the output to fit nicely onscreen. The output to `training_output.csv` is unaffected.

Prepare the tournament

```
axl.seed(1)

# players = [s() for s in axl.demo_strategies]
# players = [axl.Alternator(), axl.Defector(),
#            axl.TitForTat()]
players = [axl.Cooperator(), axl.Defector(),
           axl.TitForTat(), axl.Grudger(),
           axl.Random(), axl.Alternator()]
# players = [axl.TitForTat()]



population = dojo.Population (params_class=params_class,
                              params_kwargs=params_kwargs,
                              size = 100, #20
                              objective= objective,
                              output_filename= "training_output.csv",
                              opponents= players,
                              bottleneck= 5, #2
                              mutation_probability= 0.1, #0.1
                              print_output= False)
```

```
generations = 10 #10
results = population.run(generations)
```

```
Scoring Generation 1
     → Mean score: 2.21, Root variance: 0.19
     Generation  1 |  Best Score:  2.600000 State: 0:C:0_C_0_D:0_D_1_D:1_C_0_D:1_D_0_C
     Generation  1 | Worst Score:  1.650000 State: 0:C:0_C_1_C:0_D_0_C:1_C_0_D:1_D_0_C
Scoring Generation 2
     → Mean score: 2.36, Root variance: 0.161
     Generation  2 |  Best Score:  2.633333 State: 0:C:0_C_0_C:0_D_1_D:1_C_0_D:1_D_1_D
     Generation  2 | Worst Score:  1.916667 State: 0:C:0_C_1_C:0_D_1_D:1_C_0_D:1_D_0_C
Scoring Generation 3
     → Mean score: 2.39, Root variance: 0.148
     Generation  3 |  Best Score:  2.683333 State: 0:C:0_C_0_D:0_D_0_D:1_C_0_D:1_D_0_C
     Generation  3 | Worst Score:  1.850000 State: 0:C:0_C_0_D:0_D_0_C:1_C_1_C:1_D_1_D
Scoring Generation 4
     → Mean score: 2.4, Root variance: 0.133
     Generation  4 |  Best Score:  2.716667 State: 0:C:0_C_0_C:0_D_1_D:1_C_1_D:1_D_1_D
     Generation  4 | Worst Score:  1.983333 State: 0:C:0_C_0_D:0_D_0_C:1_C_0_D:1_D_1_D
Scoring Generation 5
     → Mean score: 2.46, Root variance: 0.192
     Generation  5 |  Best Score:  2.850000 State: 0:C:0_C_0_C:0_D_1_D:1_C_1_D:1_D_1_D
     Generation  5 | Worst Score:  1.900000 State: 0:C:0_C_0_C:0_D_0_C:1_C_1_D:1_D_0_D
Scoring Generation 6
     → Mean score: 2.44, Root variance: 0.153
     Generation  6 |  Best Score:  2.716667 State: 0:C:0_C_0_C:0_D_1_D:1_C_1_D:1_D_1_D
     Generation  6 | Worst Score:  2.033333 State: 0:C:0_C_0_C:0_D_1_D:1_C_0_C:1_D_1_C
Scoring Generation 7
     → Mean score: 2.43, Root variance: 0.12
     Generation  7 |  Best Score:  2.716667 State: 0:C:0_C_0_C:0_D_1_D:1_C_1_D:1_D_1_D
     Generation  7 | Worst Score:  2.133333 State: 0:C:0_C_0_D:0_D_1_C:1_C_0_D:1_D_1_D
Scoring Generation 8
     → Mean score: 2.43, Root variance: 0.158
     Generation  8 |  Best Score:  2.783333 State: 0:C:0_C_0_C:0_D_1_D:1_C_1_D:1_D_1_D
     Generation  8 | Worst Score:  2.000000 State: 0:C:0_C_1_D:0_D_0_C:1_C_0_D:1_D_0_D
Scoring Generation 9
     → Mean score: 2.45, Root variance: 0.15
     Generation  9 |  Best Score:  2.783333 State: 0:C:0_C_0_C:0_D_1_D:1_C_1_D:1_D_1_D
     Generation  9 | Worst Score:  1.983333 State: 0:C:0_C_1_D:0_D_1_D:1_C_1_C:1_D_0_C
Scoring Generation 10
     → Mean score: 2.44, Root variance: 0.155
     Generation  10 |  Best Score:  2.783333 State: 0:C:0_C_0_C:0_D_1_D:1_C_1_D:1_D_1_D
     Generation  10 | Worst Score:  1.800000 State: 0:C:0_C_1_D:0_D_1_C:1_C_0_C:1_D_1_C
```