User Manual

PlayMaker 8

ASDD – MSDD Learning
Reinforcement Learning
Dynamic Programming

Christopher Child, Virgile Barbieux

CITY UNIVERSITY LONDON

Contents

1. Generating Input Data

Both Rules and Maps versions are working with: ***~/InputFile.txt***

It contains X States & Actions, like this :

Input File (8 entries)	
Predator Model	**Paint Model**
W,E,E,W,E,E W,E,E,E,E,W W,E,E,W,E,N W,E,E,W,E,E W,E,E,E,E,W W,E,E,W,E,W W,E,E,W,E,W W,E,E,W,E,W	p,C,D,b,=,E p,C,d,b,-,A p,c,d,b,=,R p,c,D,b,=,R p,c,D,b,=,I p,c,D,B,=,R p,c,D,B,=,A P,c,D,B,=,A

In order to obtain these « simple » Sensors, (almost) all of the *toString()* functions were copied and modified to the *translation()* functions.

1.1 Predator Model

To generate data from the Predator Model, go to :

EnvModel.PredatorModel/PredatorTester.java

At the beggining of the file, set NUM_MOVES to the number of data wanted.

I suggest 100K for a complete simulation.

Run the file, the data set will be generated in : ***~/LogFilesResults.txt***

Copy the whole content of this one, and paste it to ***~/InputFile.txt***

The old behaviour of this file can be obtained by uncommenting line 82 :

paintEnvironment.testAgentRecords();

1.2 Paint Model

To generate data from the Paint Model, go to :

EnvModel.PaintModel/PaintTester.java

At the beggining of the file, set NUM_MOVES to the number of data wanted.

I suggest 100K for a complete simulation.

Run the file, the data set will be generated in : ***~/LogFilesResults.txt***

Copy the whole content of this one, and paste it to ***~/InputFile.txt***

The old behaviour of this file can be obtained by uncommenting line 100 :

predatorEnvironment.testAgentRecords();

2. Learn Rules

2.1 MSDD

Go to : ***V_Tester/Tester.java***

To Use MSDD, set *MSDD* (line 33) to **true**.

Set the *maxnodes* parameter (line 36) to 50.000 for a complete simulation. (The higher limit for MSDD and PreatorModel is around 23K)

To learn MSDD Rules, set *load_MSDD_rules* (line 34) to **false.**

New closedList & RuleSetList will be generated and exported to :

~/Rules.txt & ~/RSList.txt

2.2 ASDD

Go to : ***V_Tester/Tester.java***

To Use ASDD, set *ASDD* (line 40) to **true**. Also check that *MSDD* is set to **false**.

To learn ASDD Rules, set *load_ASDD_rules* (line 41) to **false.**

New closedList & RuleSetList will be generated and exported to :

~/Rules.txt & ~/RSList.txt

3. Load Rules

3.1 MSDD

Go to : ***V_Tester/Tester.java***

To Use MSDD, set *MSDD* (line 33) to **true**.

To load MSDD Rules, set *load_MSDD_rules* (line 34) to **true.**

The files ***~/Rules.txt & ~/RSList.txt*** will be imported.

3.2 ASDD

Go to : ***V_Tester/Tester.java***

To Use ASDD, set *ASDD* (line 40) to **true**.
Also check that *MSDD* is set to **false**.

To load ASDD Rules, set *load_ASDD_rules* (line 41) to **true.**

The files ***~/Rules.txt & ~/RSList.txt*** will be imported.

4. Using Maps

Go to : ***V_Tester/Tester.java***

To Use Maps, set *MSDD* (line 33) to **false** & *ASDD* (line 40) to **false**.

5. Reinforcement Learning

Go to : ***V_Tester/Tester.java***

To perform Reinforcement Learning, set *Reinforcement_Learning* (line 53) to **true**.

Set the *Reinforcement_Learning_steps* parameter (line 54) to :

- **500.000** (or more) if you're using Maps
- **20.000** if you're using Rules.

The State Generator is selected automatically, depending on whether you use Maps or Rules.

5.1 Value Table & Action Value Table

To use Value Table, set *use_Value_Table* (line 58) to **true**.

To use Action Value Table, set the above parameter to **false**.

If you use Action Value Table, you can enable Dynamic Programming by setting *use_Dynamic_Programming* (line 60) to **true**.

Here is a reminder of what both Tables look like :

Value Table	Action Value Table
STATE 1 [E, W, W, E, A, *] Value : 1.71 STATE 2 [E, E, W, W, A, *] Value : 1.66	STATE 1 [W, E, E, W, E, *] Actions : [N, E, S, W] Values : [0.58, 0.79, 0.75, 0.59] STATE 2 [W, E, E, E, E, *] Actions : [N, E, S, W] Values : [0.74, 0.71, 0.83, 0.72]

5.2 Decision Table

In order to get a much more quicker system when running the Predator App, we can convert Value & ActionValue Tables into Decision Tables.

To use Decision Tables, set *use_decision_tables* (line 55) to **true**.

Here is what a Decision Table looks like :

Decision Table
STATE 1 [W, E, E, E, E, *] Action : S STATE 2 [E, E, A, E, E, *] Action : S STATE 3 [E, E, E, A, E, *] Action : W STATE 4 [E, W, E, E, E, *] Action : W

This allows us to quickly « read » what the better action is, instead of « computing » an answer at every step.

6. Running the Predator App

Go to : ***EnvController.PredatorEnvController/PredatorEnvApp.java***

Depending on what was generated in the StateTable.txt, you should uncomment one of these lines :

- *getPredatorEnvironment().updateEnvironment()* (line 56) is the old version. It will use the default project, including Reinforcement Learning according to the last saved Percep & Actions records.

- *getPredatorEnvironment().updateEnvironmentFromStateActionValueTable()* (line 60) will import the ***StateTable.txt*** file (containing an ActionValue Table), and ask at every step for an answer from the State Generator.

- *getPredatorEnvironment().updateEnvironmentFromStateValueTable()* (line 63) will import the ***StateTable.txt*** file (containing a Value Table), and ask at every step for an answer from the State Generator.

- *getPredatorEnvironment().updateEnvironmentFromDecisionTable()* (line 67) will import the ***StateTable.txt*** file (containing a Decision Table), and directly act without any other process required. This way is a lot faster, it is the last version.

These functions can be accessed in ***EnvModel/TickEnvironment.java***

When Running the App, you'll see in the console that each Step has a number (in order to stop around 1.5K for instance, and that each Rewarded Step prints « REWARD ».

As a consequence, to get the performance of the Simulation, copy the console text into a .txt file, search for « Reward » (this will give you the number of rewarded States), and divide it by the number of steps.