HRL-MaxQ Java Code Manual

Feng Cao ::fxc100@case.edu

April 1, 2012

1 Introduction

This document describes how to use the HRL-MaxQ code, a package that includes a Java implementation of several Reinforcement Learning (RL) algorithms and typical domains (i.e. environments).

The algorithms implemented are:

- **Q Learning**: standard q learning algorithm.
- **MaxQ**: MaxQ algorithm introduced in [1], and some of variants.
- **Bayesian MaxQ**: Bayesian MaxQ algorithms proposed in our work.

The environments implemented include:

- **Taxi Domain**: As introduced in Figure 1 of [1].
- **Wargus Domain**: As introduced in [2].
- **Simple Maze Domain**: As introduced in Figure 6 of [1].
- **Hallway Domain**: As introduced in Figure 14 of [1].
- **FourDoorMaze Domain**: Incomplete, should not be used.

2 Command line format

Command line arguments are used to specify the agent and environment, as well as other parameters, to be use in the running of experiment. Figure 1 shows the usage of the command line arguments:

You could also multi-thread the experiment by using **edu.cwru.eecs.rl.MultiThreadMain**

```
Usage: java -cp bin;lib/* edu.cwru.eecs.rl.Main
-a ⟨algorithm, pr⟩              Specify the rl algorithm to be used by agent:
                                flatq | maxq | bmaxq
                                follow by pseudo reward settings:
                                none | manual | bayes | func
-e ⟨domain, param⟩              Specify the domain name:
                                taxi | hallway | simplemaze | wargus | fourdoormaze
                                followed with parameters
-exp ⟨#episode, max step⟩       Specify the number of episodes to run and max
                                step each episode
-h                              display help info
-info ⟨extra info⟩              extra information of the save to file
-run ⟨#run⟩                     multi-run
-to ⟨save_to_dir⟩               save the result to the specified directory

Examples:
java -cp bin:lib/* edu.cwru.eecs.rl.Main -a flatq -e taxi 0 -exp 1000 1000 -to res_taxi
java -cp bin:lib/* edu.cwru.eecs.rl.Main -a maxq manual -e taxi 0.15 -exp 1000 1000 -to res_taxi
java -cp bin:lib/* edu.cwru.eecs.rl.Main -a bmaxq bayes -e simplemaze 0 -exp 500 500
                                -to res_simplemaze
```

Figure 1: Usage of command line arguments

2.1 Parameter setting

There are many parameters involved here, either in the algorithm or in the environment.

- Agent parameter: (only for maxq and bmaxq)

  none: use zero pseudo reward

  manual: use manually set pseudo reward

  bayes: use bayesian pseudo reward

  func: use non-bayesian pseudo reward

- Environment parameter:

  taxi: noise

  hallway: noise

  simplemaze: noise

  wargus: arg1:config (e.g. 3322), arg2: noise

- Experiment parameter:

  arg1: number of episodes

  arg2: max step each episode

- Other parameters in different agents: (you could find them in the constructor of each agent class. They are not supposed to be specified from command line arguments. If you want to modify them, please modify the agent java files directly)

  epsilon: exploration rate – for all agents

  alpha: learning rate – for all agents

  iter_sim: number of episodes per simulation – for Bayesian MaxQ agents

  maxStep_sim: max step each simulation episode – for Bayesian MaxQ agents

  epsilon_sim: exploration rate in the simulation – for Bayesian MaxQ agents

  alpha_sim: learning rate in the simulation – for Bayesian MaxQ agents

3 How to implement your own Agent and Environment

3.1 Implementing Agent

All the related java files are under package *edu.cwru.eecs.rl.agent*. One could refer to *FlatQAgent.java* for a sample implementation of agent. Basically, in order to add a new agent, say *NewAgent*, to the package, *NewAgent* has to extends *Agent* or its subclass. Also, *AgentType.java* defines different types of agent. So you may probably want to add your own type into this enum. The reason why we use an *AgentType* is that each implementation of Agent, say MaxQAgent, may contain agents with different behaviors. Thus the member variable *agentType* helps decide what behavior is expected.

3.2 Implementing Environment

All the related java files are under package *edu.cwru.eecs.rl.env*. One could refer to *TaxiEnv.java* for a sample implementation of environment. Basically, in order to add a new environment, say *NewEnv*, to the package, *NewEnv* has to extends *Environment* or its subclass. Also,

References

[1] Thomas G. Dietterich. Hierarchical reinforcement learning with the maxq value function decomposition. *Journal of Artificial Intelligence Research*, 13:227–303, 2000.

[2] N. Mehta, S. Ray, P. Tadepalli, and T. Dietterich. Automatic discovery and transfer of MAXQ hierarchies. In Andrew McCallum and Sam Roweis, editors, *Proceedings of the 25th International Conference on Machine Learning*, pages 648–655. Omnipress, 2008.