Minefield Navigation Simulator (2008)

Authors: Ah-Hwee Tan, Dan Xiao, Ning Lu, Jun Jin
School of Computer Engineering
Nanyang Technological University
Nanyang Avenue, Singapore 639798

Documented: April 2008

This software contains the implementation of a number of reinforcement learning systems, including R-FALCON, TD-FALCON, and backpropagation neural network with Q-learning, on a minefield navigation task. You are free to use and modify the codes for your own work. However, please understand that the source codes are provided on a "as is" basis. In other word, no technical support can be provided due to the lack of manpower resources.

We hope the simulator will help in your research. If you make use of this work, please cite the following references:

- Ah-Hwee Tan, Ning Lu and Dan Xiao. Integrating Temporal Difference Methods and Self-Organizing Neural Networks for Reinforcement Learning with Delayed Evaluative Feedback. *IEEE Transactions on Neural Networks*, Vol. 9, No. 2 (February 2008), 230-244.

- Dan Xiao and Ah-Hwee Tan. Self-Organizing Neural Architectures and Cooperative Learning in Multi-Agent Environment. *IEEE Transactions on Systems, Man, and Cybernetics - Part B*, Vol. 37, No. 6 (December 2007), 1567-1580.

Key Panels

The minefield simulator consists of four display panels, including Minefield (View from the top), Sonar Signal Input, AV Sonar Signal, Current and Target Bearings, as well as two control panels, namely Automatic Control and Manual Control.

The **Minefield panel** displays the two-dimensional minefield configuration, showing the position of the agent (Green Tank), the target (Red Flag), and the mines (Black Bombs).

The **Sonar Signal panel** shows the strength of the sonar signals received by the agent in the five directions for detecting mines and boundaries of the minefield.

The **AV Sonar Signal panel** shows the strength of the sonar signals received by the agent in the five directions for detecting **other agents** and boundaries of the minefield.

The **Bearing panel** shows the bearing of the agent as well as the bearing of the target relative to the bearing of the agent.

The **Manual Control panel** is for a human operator to control the agent towards the target using the five directional buttons: <, left, Ahead, Right, and >.

The **Automatic Control panel** is for conducting experiments, wherein the agent learns to perform the task of navigation through reinforcement learning. There are three key functions under Automatic Control.

1. **Reset**: Randomly generate a new minefield configuration
2. **Step**: Run through the trial step by step to see how the agent navigates towards the target.
3. **Auto**: Conduct single or multiple experiments to evaluate the performance of the agent across learning trials. The learning progress and performance are shown on the DOS window and tabulated in the output file (result.txt) after the experiments complete.

Key Parameter Settings

The following is a list of the key parameters that can be set at the Automatic Control interface.

Experiment Settings

Agents: Number of Agents, default is 1
Steps: The maximum number of steps the agent can take in each trial, default is 30
Trials: Number of experiment trials, default is 3000
Target Moving: if not ticked, the target is stationary

View Parameters

Interval: Time delay to slow down the simulation. Use in Auto mode to show the navigation process.
Track: Show the navigation path of the agent. Use in Auto mode.

Learning Parameters

Immediate reward: If ticked, immediate reward is provided. Else, delayed reward is used.
Bounded TD Rule: If ticked, Bound TD formula is used. Else, the original (unbounded) version is used.

Key Parameter Changeable in the Source Codes

The simulator can be run directly without modifying the parameters in the interface or source codes. As in MNT.java, the default setting of the simulator is as follows

```
// Important RunTime Parameters

private static int      TDMethod =QLEARNING;
private static int      AVTYPE    =TDFALCON;
private static int      numRuns =1;      // Number of complete simulation runs
private static int      interval =100;   // Interval of learning trials for tabulating results
```

To begin, just click on Auto to see a sample run of TD-FALCON.

1. Changing TDMethod and Agent Type

The TDMethod can be changed from QLEARNING to SARSA.
The agent type may be changed to from TD-FALCON to RFALCON, or BPN.
However, different systems require different training schedule and thus different parameter settings.

Referring to AGENT.java, the QEpsilonDecay rate has to be changed.

```
// default QEpsilonDecay and initialQEpsilon for TD-FALCON
// The values are automatically overridden in the setParameters method when using RFALCON, or BPN.

public static double QEpsilonDecay = (double)0.00050;
public static double QEpsilon          = (double)0.50000;
```

For display purpose, interval may be increased as well for longer learning schedules.

2. Changing Reward Scheme

The reward scheme can be set during runtime on the simulator. However, RL systems work better with specific parameters for different reward scheme. Referring to AGENT.java again, to switch the reward scheme between Delayed and Immediate, QGama has to be changed accordingly.

```
// default QGamma suitable for Delayed Reward Scheme
// The value is automatically overridden in the setParameters method when using immediate reward.

public static double QGamma    = (double)0.9;
```