Game Reinforcement Learning

Pranav Maneriker, Arnab Ghosh , Arpit Shrivastava , Piyush P. Kurur

Computer Science and Engineering , IIT Kanpur

Objectives

The primary objectives in our project were:

- Create a library for Reinforcement Learning
- Primary Reinforcement Learning algorithms such as Q-Learning and SARSA had to be written
- Create an interface for learning the best moves for a 2 player game
- Provide some example 2 player games that conforms to the above standard and test the library.

Introduction

The Machine Learning library in Haskell (HLearn) lacked a support for Reinforcement Learning Algorithms.Hence we started off by implementing the primary set of Reinforcement Learning Algorithms : Q-Learning and SARSA which are the 2 most frequently used Reinforcement Learning algorithms .
To test our algorithms we implemented a small game in our system where a cat and mouse are placed on a board with some obstacles and a piece of cheese and the objective of the mouse is to get the cheese and the objective of the cat is to get the mouse .
We then provide an interface to specify any 2 player game in terms of its valid set of moves and a reward function associated with each state which greatly simplifies anyone who wants to write a 2 player game using the reinforcement learning algorithms supported by our library.

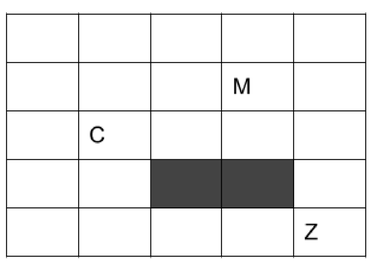

Figure 1: Cat and Mouse Game

Reinforcement Learning

Reinforcement learning is an area of machine learning inspired by behaviorist psychology, concerned with how software agents ought to take actions in an environment so as to maximize some notion of cumulative reward.[1]
The basic reinforcement learning model consists of:[1]

1. a set of environment states S;
2. a set of actions A;
3. rules of transitioning between states;
4. rules that determine the scalar immediate reward of a transition;
5. rules that describe what the agent observes.

Crux of Reinforcement Learning

A reinforcement learning agent interacts with its environment in discrete time steps. At each time t, the agent receives an observation o_t, which typically includes the reward r_t. It then chooses an action a_t from the set of actions available, which is subsequently sent to the environment. The environment moves to a new state s_{t+1} and the reward r_{t+1} associated with the transition (s_t, a_t, s_{t+1}) is determined. The goal of a reinforcement learning agent is to collect as much reward as possible.[1]

SARSA

The major steps of the algorithm are : [2]

1. Initialize the Q-values table, Q(s, a).
2. At a particular state s choose an action, a, based on an ϵ parameter i.e with ϵ probability a random action is chosen or else the action with the maximum Q value is chosen .
3. Take the action, and observe the reward, r, as well as the new state, s'.
4. Update the Q-value for the state using the observed reward and the maximum reward possible for the next state.
5. Set the state to the new state, and repeat the process until a terminal state is reached.

QLearn

The major steps of the algorithm are [2]:

1. Initialize the Q-values table, Q(s, a).
2. At a particular state s choose an action, a, based on an ϵ parameter i.e with ϵ probability a random action is chosen or else the action with the maximum Q value is chosen .
3. Take the action, and observe the reward, r, as well as the new state, s'.
4. Update the Q-value for the state using the observed reward and the maximum reward possible for the next state.
5. Set the state to the new state, and repeat the process until a terminal state is reached.

Update Equations

The update equation for QLearn is : [2]
$Q(s,a) \leftarrow Q(s,a)+\alpha[r+\gamma max_{\alpha'} Q(s',a')-Q(s,a)]$
The update equation for SARSA is : [2]
$Q(s,a) \leftarrow Q(s,a)+\alpha[r+\gamma max_{\alpha'} Q(s',a')-Q(s,a)]$
The major difference between it and Q-Learning, is that the maximum reward for the next state is not necessarily used for updating the Q-values. Instead, a new action, and therefore reward, is selected using the same policy that determined the original action. Another difference between the 2 algorithms is the fact that in SARSA 2 action steps are required to determine the next state-action pair along with the first.

Implementation

```haskell
data Game state action = Game{
  isTermState :: state -> Bool,
  reward :: state -> (Double, Double),
  nextState :: state -> (action, action) -> state,
  getPossibleActions :: state -> ([action],[action]),
  startState :: state,
  trainer :: state -> (action, action)
  }
```

Figure 2: The Game ADT

Available module functions :

- `Learn.Learner.learnGame`
- `Learn.Helpers.getAction`
- `Game.playGameInteractive`

References

[1] Wikipedia.
Reinforcement learning.
Retrieved from Wikipedia on 13 November 2015.

[2] Richard S. Sutton and Andrew G. Barto.
Reinforcement learning: An introduction.

Acknowledgements

We are thankful to the course Functional Programming and especially Professor Piyush Kurur for providing us this opportunity to work in this field and also we thank the Haskell wiki which accurately states the usage of several functions without which it would have been much more difficult.

Contact Information

- Web: https://github.com/arnabgho/RLearnHaskell
- Email: mailto:mpranav@iitk.ac.in
- Email: mailto:arnabgho@iitk.ac.in
- Email: mailto:shriap@iitk.ac.in