This repository Search Pull requests Issues Marketplace Gist

rodmcnew / **tabular-sarsa-js**

Watch 1 Star 1 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Insights

A reinforcement learning agent for environments with discrete states

54 commits **1** branch **5** releases **1** contributor MIT

Branch: master New pull request Create new file Upload files Find file Clone or download

rodmcnew committed on **GitHub** Update README.md Latest commit `5058e78` 23 days ago

example	cleanup code and docs	a month ago
src	convert to ES5	24 days ago
.gitignore	add files	a month ago
LICENSE.txt	add files	a month ago
README.md	Update README.md	23 days ago
package.json	convert to ES5	24 days ago

README.md

Tabular Expected SARSA Agent

This contains an agent that learns to maximize reward through reinforcement learning. The agent works by building a table that can predict the expected value of every possible action from every possible state. Exploration is accomplished by following an epsilon greedy policy.

Because this uses a table-based Q function, it only works in environments with a discrete set of states and actions. You must be able to convert all states and actions to integers to use this agent.

Installation:

```
npm install tabular-sarsa
```

Usage:

```
var agent = new tabularSarsa.Agent(
    numberOfPossibleStates,
    numberOfPossibleActions
);
var lastReward = null;

function tick() {
    /*
     * Tell the agent about the current environment state and
     * have it choose an action to take.
     */
    var action = agent.decide(
        lastReward,
        environment.getCurrentState()
    );

    /*
     * Take the action inside the environment find out how
     * rewarding the action was.
     */
    lastReward = environment.takeAction(action);
}
```

Saving trained agents for later:

```
//Saving an agent
var agentA = new tabularSarsa.Agent(100, 4);
var savedAgentData = agentA.saveToJson();

//Loading an agent
var agentB = new tabularSarsa.Agent(100, 4);
agentB.loadFromJson(savedAgentData);
```

Extra options:

```
var agent = new tabularSarsa.Agent(
    100,//Number of possible states
    4,//Number of possible actions
    {
        learningEnabled: true,//set to false to disable all learning for higher execution speeds
        learningRate: 0.1,//alpha - how much new experiences overwrite previous ones
        explorationProbability: 0.05,//epsilon - the probability of taking random actions in the Epsilon Gr
        discountFactor: 0.9,//discountFactor - future rewards are multiplied by this
    }
);
```

Optimizations beyond plain SARSA that speed up learning:

- Uses "Expected SARSA" rather than plain SARSA
- Uses the first seen reward for each state-action as the initial Q value

More info about the Expected-SARSA algorithm: http://www.cs.ox.ac.uk/people/shimon.whiteson/pubs/vanseijenadprl09.pdf