Robot Learning Project 4 ENPM 808F

Dots and boxes reinforcement learning

By-

Arpit Maclay(116314492)

Aim-

To train a 2x2 and3x3 game of dots and boxes using reinforcement learning.

Methods-

1.Q-learning- A Q learning algorithm is initiated by first making a Q table which contains states and actions as its row and columns such that with every action the values are checked according to the cost and the values keeps updating in the q table which can be said the training of the system.

2. Functional approximation- Any function can be converted into a neural network.

Also, In cases where there are plethora of states q matrix cannot handle so many values and may fail in some cases such that neural networks are used which can handle much more states and their calculations at the same time.

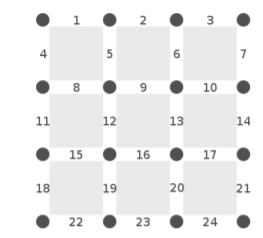
Strategy-

State representation- The state is represented as one instance of the game and the actions are the moves or lines that are placed one by one.

For simplification one state is converted into a binary form where a filled line means ‘1’ and a void means a ‘0’.

Such that for 12 lines we have 2^11 states.

State matrix- number of states×max(number of actions per state)



Feedback-An action generates feedback from the environment. The feedback can be either positive or

negative. The feedback is used in the Q-learning algorithm for estimating how good an action

is. The term reward is used for positive feedback and negative feedback is called punishment.

Learning Rate-The learning rate is a parameter in the Q-learning algorithm, describing the relationship between

old and new memories. A high value means that new memories take precedence over old

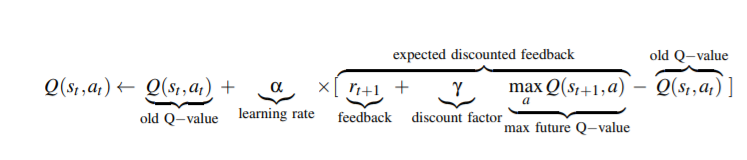
memories and vice versa. A value of zero means that nothing will be learnt.

Discount factor- The discount factor is a parameter in the Q-learning algorithm, affecting the importance of future feedback. A high discount factor increases the importance of future feedback.

Approach-

The simplest form of Q-learning stores a value for each state-action pair in a matrix or table. The fact that it requires one value for every state-action pair is one of the drawbacks of Q learning; it requires a huge amount of memory.

The algorithm for updating Q-values is shown in equation 1,



If there is one, and only one, stable solution, that is which action is the best in each state, the

Q-learning algorithm will converge towards that solution. Because of the back-propagation

property of Q-learning, this however requires a large number of visits to every possible state action pair.

Q Table-

The Q table has the all the states and the weights associated with possible actions of that state. The simple way to implement the qtable was to have a a numpy array with the number of rows as the number of all possible states and the number of columns as number of all possible actions in a state. While this was feasible for a 2x2 game. The number of states in a 3x3 game is 16777216. Maintaining a numpy array of this size was very inefficient. So instead the q table was represented by a dictionary, to which each state was the algorithm comes across was added.

Agents used to train-

Q agent - The Q-learning agent uses the Q-learning algorithm to choose an action for a given state.

Random agent- The purpose of the random agent is to be unpredictable. It does always perform a random action upon any given state. This agent is used to train the Q-learning agent in as many states as possible, that is generate lots of training data.

Tests-

The tests consist of first letting the game play itself through Q learning storing the Q values. Training it for 100,1000 and 10000 iterations. Then it is played a random player and the output is recorded.

Same tests procedure will be carried out for functional approximated neural network.

Testing

In this section results obtained from the tests executed during the writing of this report are presented.

**2X2 Grid**

For 100 Training Games-

QAgent win count – 38 wins

Random Agent win count – 40 wins

Ratio of Q agent to random agent – 0.95

Observation- The Q learning agent is not better than the random agent.

For 1000 Training Games-

Q Agent win count- 420

Random agent win count-410

Ratio of Q agent to random agent-1.024

Observation- The Q agent is better than random agent but by very small amount.

For 10000 Training Games-

Q Agent win count –8074

Random agent wins count-1598

Ratio of Q agent to random agent-5.05

Observation- The Q agent is performing much better than the random agent.

**3X3 Grid**

For 100 Training Games-

Q Agent win count –51

Random agent wins count-49

Ratio of Q agent to random agent-1.04

Observation- The Q agent is performing not any better than the random agent.

For 1000 Training Games-

Q Agent win count- 531

Random agent win count-469

Ratio of Q agent to random agent-1.13

Observation- The Q agent is little better than random agent.

For 10000 Training Games-

Q Agent win count- 8023

Random agent wins count-1977

Ratio of Q agent to random agent-4.05

Observation- The Q agent is much better than the random agent.