Battlesnake using Deep Q Learning

Petter Amundsen , Tommy Bergsvåg and Håkon Sagehaug

Bouvet

Objectives

Make Deep Q learning neural network for training snakes used for playing the game Battlesnake

- Can we use Deep Q learning for training snakes to play the game
- Can we make the AI snakes perform better than traditionally programmed snakes

Introduction

Poster for describing the work done for student project in INF626. We wanted to see if we could use Deep Q learning for training snakes for playing the game Battlesnake. Battlesnake is a game played on a squared board. For training we used an 11x11 board. The goal of the game is to survive. For a snake to survive it must; eat, not crash into other snakes and not crash into the wall - the last snake on the board is the winner.

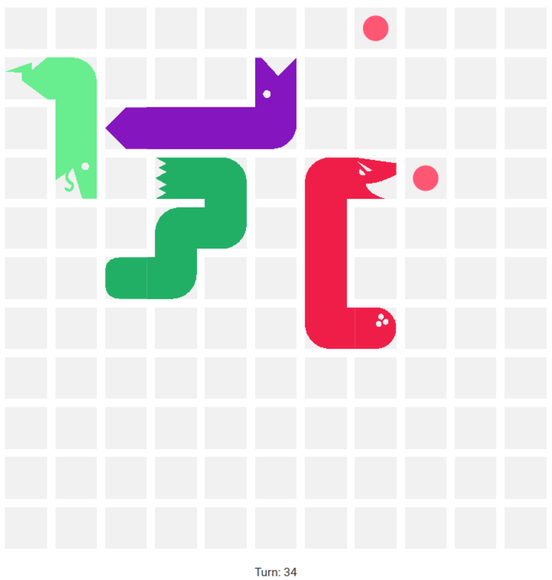

Figure 1:Battlesnakes game: 4 snakes compete in surviving the longest.

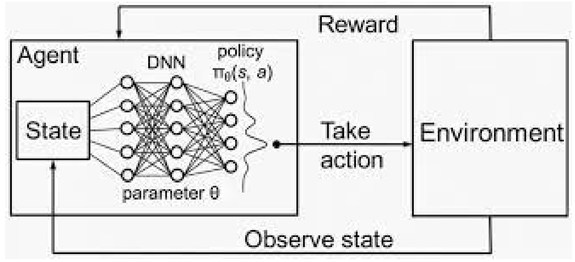

Figure 2:Overview of deep Q network.

Method

Before each move the state of the game is fed to the agent as an 11x11 array, holding the coordinates of every snake head and body, as well as food. Before the network is properly trained it should do some exploration, trying random moves, so for every move there is a ϵ probability of the agent doing something at random. ϵ decays over time at an exponential rate, converging towards 0. For every action, there is a reward, and the neural network will adopt the weights that maximizes reward. The snakes have been given different reward functions for trying out different strategies. The neural network is implemented with Keras/Tensorflow.

Results

Figure 3:Average survival measured in number of turns.

Figure 4:Probability of winning for each snake, as the training progressed.

All the snakes performed better over time, but the snake named 'survivor' outperformed the others significantly. Additional training after the first 10000 games did not appear to increase performance.

Implementation/Method

Trained three different snakes each having a reward function

1. Snake only gets one point if is the only snake left meaning it won the game
2. Snake is punished when loosing the game, and rewarded when winning
3. Snake gets point if it survives

Then we created three different models using Keras Sequential model with input

1. Shape of the board(11x11)
2. Activation function, here we used ReLU and LeakeyReLU
3. One model with 2 layers and two with three layers

Discussion

Some bugs were discovered in the game code that prevented the snakes from learning. These issues were hard to troubleshoot and were discovered mostly by coincidence. There might be other such bugs lurking, preventing the snakes from performing at a superhuman level. It is also possible that the top snake is being held back by the other low performing snakes, as the game ends when there is just one snake left on the board. If we had replaced all the snakes with four identical 'survivor' snakes, we would likely have seen longer lasting games.

Conclusion

We set out to use reinforcement learning to train battlesnakes. The snakes did not perform at the expected level, but were much better than random chance.

Additional Information

Source code can be found on github: https://github.com/petteramu/battlesnakes-dqn

Contact Information

- Petter Amundsen, petter.amundsen@bouvet.no
- Håkon Sagehaug, hakon.sagehaug@bouvet.no
- Tommy Bergsvåg, tommy.bergsvag@bouvet.no