Exploration of Reinforcement Learning to SNAKE

Bowei Ma, Meng Tang, Jun Zhang

CS229 Machine Learning, Stanford University

Abstract

Reinforcement learning is an essential way to address problems where there's no single correct solution. In this project, We combined convolutional neural network and reinforcement learning to train an agent to play the game Snake. The challenge is that the size of state space is extremely huge due to the fact that position of the snake affects the training results directly while it's changing all the time. By training the agent in a reduced state space, we showed the comparisons among different reinforcement learning algorithms and approximation optimal solution.

Methods

1. Mathematical Analysis

- Snake Game is to find a self-avoiding walk(SAW) in R^2
- Picking the shortest SAW for each step is a NP-HARD problem

2. Approximated Optimal Solution

- Find the shortest path from head to food
- Guarantee the existence of path from head to tail after absorbing the food.
- Otherwise follow the longest path from head to tail

3. Q-learning

- Train agent to learn optimal policy from history of interaction with environment. History is a sequence of state-action-rewards$< s_0, a_0, r_1, s_1, a_1, r_2, s_2, a_2, r_3, \ldots >$
- Off-policy Learning: Use Bellman equation as an iterative update

$$Q_{i+1}(s,a) := \mathrm{E}_{s' \sim \varepsilon}\{r + \gamma \max_{a'} Q_i(s', a'; s, a)\}$$

- Use neural network to approximate value of Q-function by using

$$Q(s,a) := Q(s,a) + \alpha(r + \max_{a'} Q(s', a') - Q(s,a))$$

State	eat food	hit wall	hit snake	else
reward	+500	-100	-100	-10

4. SARSA (State-Action-Reward-State-Action)

- On-policy Learning: Along exploration, the agent iteratively approximates the value of a policy, and takes action follow that policy.

$$Q(s_t, a_t) := Q(s_t, a_t) + \alpha(r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t))$$

Evaluation

1. Learning Curves

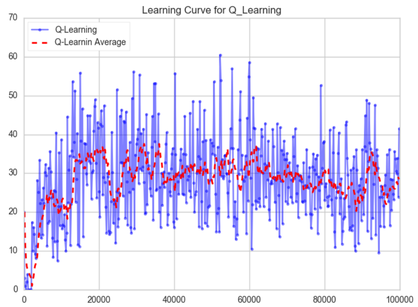

Figure 1

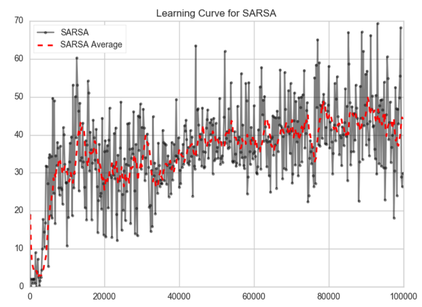

Figure 2

- The learning curves from Q-learning (figure1) and SARSA(figure 2) are shown above respectively.
- **Finding**: Q-learning improves performance with fewer number of trials, but in long-run the performance are not guaranteed to b improving.

2. Performance Comparison

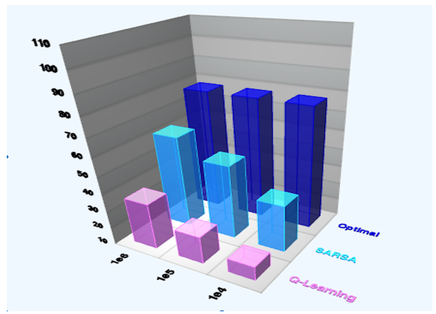

Figure3

# Iter	Opt	SARSA	Q-L
1e+04	77.504	36.858	18.023
1e+05	77.504	51.994	25.789
1e+06	77.504	61.830	36.567

Figure4

- **Figure 3 & Figure 4** displayed the average scores achieved in 1-minute time limit by agent with Optimization Algorithm, agent trained in SARSA and agent trained in Q-Learning respectively.
- **Finding**: Agent trained in SARSA performed better than the one with Q-Learning significantly. After 1e+06 iteration, the former could achieve 80 percent of performance of Optimization Agent.

Observations & Analysis

- Using quadrant mapping of the state space, the size of state space is tremendously reduced and learning rate has been accelerated.
- SARSA seems to outperform Q-Learning in long run, but also reveals a sluggish learning curve.
- Q-Learning would reinforce its self-righteous Q-value even with small learning rate, and thus leads to considerably volatile performance.

Discussion

- Even with decreasing exploration probability, the Q-Learning is not stable.
- Other approximation of the state space should be explored for better performance.
- Various turning parameters should be implemented to improve the probability of convergence for Q-Learning.

Related Work

1. Risto Miikkulainen, Bobby Bryant, Ryan Cornelius, Igor Karpov, Kenneth Stanley, and Chern Han Yong. 2006. Computational Intelligence in Games.
2. Giovanni Viglietta. 2013. Gaming is a hard job, but someone has to do it!

Stanford University