Game Agent

Ezri Y. - 311599039,Or L. - 302618103,
Tikva s. – 311550776, Adiram s. – 204040471

January 2018

1 Abstract

In this paper, will present how to teach an agent to play a game from Open AI by three methods. Firstly, a small amount of data (finite) will be answered by Q-table that represent for each step where is the best way to go. Secondly, an infinite amount of data will be answered by a linear regression that will bring the best-predicted action. Lastly, another way to resolve an infinite amount of data will be by a deep neural network (DNN) that gives the best-predicted action to do.

2 Introduction

Nowadays the technology is developing very fast and with this improvement, there is need to improve our lives with the help of machines. The demand for a machine that learns the environment that can replace the fighter soldier in the battle or the tired drivers on the road is become to be more necessary. our project represents the ability of an agent to perform the best act in a dynamic basic environment. using those principles can help the industry to create robots more complicated and sophisticated that can to deal with more complex environment as the real world.

The purpose of this project is to make an agent that know how to handle his actions dynamically in the given environment. The agent use Reinforcement Learning algorithms to handle the environment. Reinforcement Learning is a technique of machine learning that based on the behaviorist psychology method, that calls to focus on trial and error between the agent to the environment.

The environment of the agent is dynamic with his own rules e.g. it can be finite or infinite with certain limitations. Every action of the environment of the agent makes change's in the environment that brings the agent to deal with a new situation and threats. For any action that the agent makes he gets a reward for better or worse.

3 Previous attempts

At first we train an agent to play Frozen-Lake game. In this game the environment is grid 4*4 of four states: S: starting point, safe. F: frozen surface, safe. H: hole, fall to your doom and lose in game. G: goal, win game The purpose of this game is to get G state without step on H state. To solve this problem, we made a Q-table in size 4*16 (4 actions: up, down, left, right in 16 different states), that consists Q-values. Q-value is a tuple of (state, action) that brings the optimal chosen action (action with max reward from given state). each cell (Q-value) update by the formula of Q-learning :

$$\underbrace{Q(s_t, a_t) \leftarrow Q(s_t, a_t)}_{\text{The New Action Value = The Old Value}} + \underbrace{\alpha}_{\text{The Learning Rate}} \times \Big(\underbrace{r_{t+1} + \lambda \max_a Q(s_{t+1}, a)}_{\text{The New Information}} - \underbrace{Q(s_t, a_t)}_{\text{the Old Information}} \Big)$$

Figure 1: Q-learning formula

After training, the agent uses the Q-table to choose the best action to go in each state. The above works fine for small amount of data. How can we deal with large number of states? We Turn the Q-values into a Regression Problem. That is, we try to predict the Q-values (for each action) given the state (we make it on a small amount of states to show ability)
1. At first we will do random action no matter what it will be just to get the initialize step.
2. Choose action to move the next state by Exploration (random) or Exploitation (take the maximize Q-value action)
3. Get a new state by this action and by this action update the true Q-value like we explain before.
4. UPDATE the loss function
5. UPDATE the weights with the new loss. From now on the agent will decide his steps strategy by max predict Q-value on actions.

4 required background - The cart Pole Game:

A pole is attached by an un-actuated joint to a cart, which moves along on a track. The system is controlled by applying a force of LEFT or RIGHT to the cart. The pole starts upright, and the goal is to prevent it from falling over. A reward of +1 is provided for every timestamp that the pole remains upright. The episode ends when the pole is more than 15 degrees from vertical, or the cart moves more than 2.4 units from the center or the game achieved 200 score (winning in the game).

5 Project Description:

In the cartPole game, there are a lot of different observations (world situations) in each game. The cartPole problem can be modeled by the Deep neuronal network (DNN). DNN - is an Artificial neural networks (ANN) with multiple hidden layers between the input and output layers.

5.1 Make data for training

To make a lot of good data for training the agent, let denote Q-value as a tuple of:
1. Action - chosen by random from all the actions that the agent could take.
2. Previous-observation - an environment-specific object representing the environment.
The tuples collected per game and the scores of them sum up. When the game will finish, the reinforcement learning comes to play and decide to the agent if this score is good enough for training by a threshold that the programmer decide to the agent from the beginning.

5.2 Build the deep neural network

The network the agent learned from is a deep neural network. The network has seven layers: The first layer is input layer, that gets observations. Five hidden layers, with different size 128 256 512 256 128 Respectively. Each hidden layer use releu, rectified linear unit - choose the max between 0 to given x, function activity. Between the hidden layers, there is dropout function that creates regularization to prevent overfitting problem. the dropout decides how much neurons can activate from all that activated. The output layer consists the number of the actions that can take in the specific environment in this case - tow actions. The activation of this layer is softmax.

5.3 training the network

the input is observations and the label is an action.
The learning is done by the training that builds before

5.4 Using the model

The agent selected his actions by the DNN model.
The testing had tested on 30% of the chosen num of games.

6 experiments/simulation results:

6.1 Running the game randomly:

Average: 22.042666666666666
Median : 19.0
counter : Counter(15.0: 111, 16.0: 92, 13.0: 89, 14.0: 89, 17.0: 78, 12.0: 77, 18.0: 64, 11.0: 60, 19.0: 60, 20.0: 56, 10.0: 55, 25.0: 54, 22.0: 49, 24.0: 45, 21.0: 45, 23.0: 38, 26.0: 35, 9.0: 29, 28.0: 27, 29.0: 27, 27.0: 26, 33.0: 26, 30.0: 23, 32.0: 23, 35.0: 21, 31.0: 19, 34.0: 16, 36.0: 14, 38.0: 13, 37.0: 12, 41.0: 10, 42.0: 10, 44.0: 10, 40.0: 10, 43.0: 9, 39.0: 8, 55.0: 6, 60.0: 5, 50.0: 5, 52.0: 5, 51.0: 4, 8.0: 4, 48.0: 3, 47.0: 3, 45.0: 3, 68.0: 3, 49.0: 3, 58.0: 2, 62.0: 2, 75.0: 2, 59.0: 2, 54.0: 2, 70.0: 1, 96.0: 1, 94.0: 1, 64.0: 1, 76.0: 1, 46.0: 1, 61.0: 1, 69.0: 1, 56.0: 1, 142.0: 1, 53.0: 1, 71.0: 1, 72.0: 1, 65.0: 1, 89.0: 1, 63.0: 1)

6.2 Running the game by the model:

The number of games is 5000 divided to: 70% games to testing (3500 iterations)
30% games of testing (1500 iterations)

6.3 The training data results

INFO:log file:Average: 60.11965811965812
INFO:log file:Median : 57.0
INFO:log file:counter : Counter(50.0: 11, 54.0: 8, 51.0: 8, 52.0: 8, 63.0: 8, 56.0: 8, 58.0: 7, 57.0: 7, 75.0: 6, 53.0: 5, 55.0: 5, 60.0: 5, 71.0: 3, 67.0: 3, 66.0: 3, 59.0: 3, 65.0: 3, 68.0: 3, 70.0: 2, 76.0: 1, 90.0: 1, 78.0: 1, 99.0: 1, 79.0: 1, 62.0: 1, 82.0: 1, 61.0: 1, 84.0: 1, 73.0: 1, 69.0: 1)

6.4 Training the model the results :

The loss function: 0.66416
The accuracy: 0.6050

6.5 The testing that works with the DNN model results:

2018-01-15 19:02:23,875 - INFO - scores: [200.0: 1500]
2018-01-15 19:02:23,886 - INFO - Average Score:200.0

7 Conclusion

In conclusion, we have provided some uses of reinforcement learning by 3 methodology - Q-table, linear regression a deep neural network. For each methodology,

the RL reflected by a different method. Q-table - suitable to finite data and not efficient on a big amount of data (can't be used in infinite amounts). The Q-tables works with RL by a Q-learning formula to give the best-expected reward in each step. linear regression - suitable for infinite amounts of data work with RL by a Q-learning formula to get the label. DNN - suitable for infinite amounts of data work with RL by giving trail as a threshold.

References

[1] Michael L. Littman Leslie Pack Kaelbling and Andrew W. Moore. Reinforcement Learning: A Survey . on central issues of reinforcement learning form this article we will learn and understand in widely and deeply way how reinforcement learning actually work to maximize his offers on our agent. *Journal of Articial Intelligence Research*, 322(10):237–285, 1996.

[2] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, and Martin Riedmiller Daan Wierstra. Playing Atari with Deep Reinforcement Learning. this paper demonstrates that a conventional neural network can overcome some challenges that the other deep learning applications cant deal with, as play and perform good actions in playing games. the results show that the agent had learned so well that he got a better score than humans. it seems that that in this paper they did what we try to do at the first part of our project, so it can be very useful. *deepmind.com*, 322(10):1–14, 2013.

[3] Alexander Pritzel Nicolas Heess Tom Erez Yuval Tassa David Silver Timothy P. Lillicrap, Jonathan J. Hunt and Daan Wierstra. CONTINUOUS CONTROL WITH DEEP REINFORCEMENT LEARNING Approximation - . at this paper presents some problems with the dqn learning technique for teaching agent to play games. and how they solve them. maybe we will try to use this technique to improve our algorithm. *Published as a conference paper at ICLR 2016*, 322(10):1–14, 2016.