Machine Learning Engineer Nanodegree
Capstone Proposal

Markus Buchholz
27/07/2017

DQN algorithm in Reinforcement Learning control system
– Proposal –

Domain Background

These project discuses reinforcement learning approach to control CartPole. Research focuses on implementation of deep neural network to control game (CartPole) and hyper parameters tuning in order to achieve in shortest possible way the defined goal (reward of 195 over 100 consecutive trials).
In 2013 DeepMind company published its breakthrough paper: *Playing Atari with Deep Reinforcement Learning* [1], [2]. The paper demonstrates achievements in implementation of pioneering new algorithm called **Deep Q Network (DQN)** used to learn the computer to play Atari 2600 video games. The result of presented work (control approach), where the pixel screens were observed (environment state), reward given and control action for the game taken was outstanding. The main concept of DQN is to replaces the Q-table in ordinary Q-Learning algorithm by deep neural network to approximate the action reward based on the state.
Artificial intelligence has become the most modern and powerful technology in solving human being challenges and to control robots. It many cases advantages of using this approach is highly relevant with proper algorithm choice in order to solve mathematical problems, secure precise control and less computing time. Deep Learning Nonoprogram I am a student at and previous SmartCab project where the Reinforcement Learning and Q algorithm was investigated inspired me to look closer these technologies and study deeper presented approach.

Problem Statement

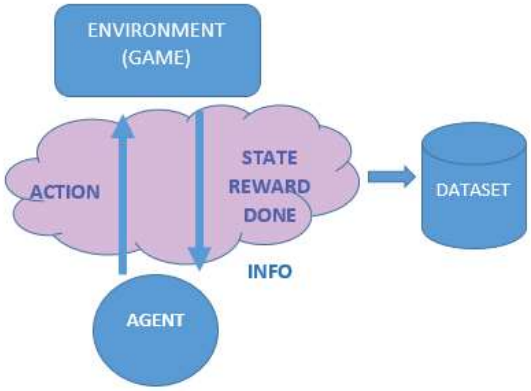
The problem, which is going to be solved, is based on relatively simple environment in OpenAI gym (a game simulator). CartPole is a cart (moving on frictionless track) with the attached by an un-actuated joint pole. The cart moves (left – right) preventing from the pole falling over. The game agent (control systems) is rewarded +1 for every time step that the pole remains upright.
In this research there is a necessity to design automatic (intelligent) system to control (balance/prevent from falling over) the pole for the defined time (the problem is solved if agent receives average 195.0 score over 100 consecutive trials). The control system will be based on DQN.

Datasets and Inputs

In this project, the dataset is not provided as an existed data base but rather the data is collected during the training process (playing the game).
During the game, while the agent trying to balance the CartPole by giving the action (agent_action - agent by giving the command 0 or 1 moves the cart to left or right) the environment sends to agent a “tensor” including following (below) information. The set of states and actions, together with rules for transitioning from one state to another, make up a Markov decision process.

next_state, reward, done, _ = env.step (agent_action)

1. **next_state** – consisting of information about the position if the cart on the track; angle of the pole to vertical line; linear velocity of the cart and angle velocity of the pole).
2. **reward** – Every time (frame) the agent "balanced" the pole (less than 15 degrees) the reward +1. In this project the target is a score of 195.
3. **done** - is a Boolean value telling whether the game ended or not.
4. _ - info (nor relevant in this project)

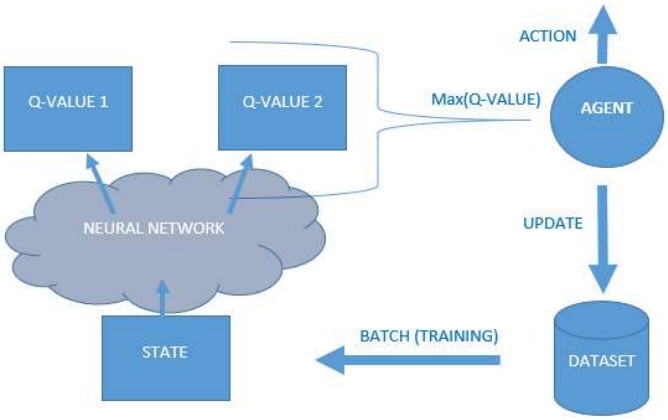


In order to train the neural network (to control the cartpole) the dataset has to be provided. Here for each step of the game (time frame) previous state of the game (state of cartpole) is collected. Random sample of these information (batch of previous states) learn the network (adjust the weights).

Solution Statement

In this project, the goal is to secure the minimum score of the game. When the reinforcement learning is applied to solve the problem with finite states of environment so the Q – learning algorithm based on discrete Q table can be successfully used. In this project case, the state space is continuous, therefore the simple tabular representation will not work. In order to approximate arbitrary the continuous function, the neural networks have to be used. The approximate Q function where $\overline{w}$ is a vector of weights of neural network, which will be optimized during the training process. Finally, the output from the iterative weight optimization process – trained neural network, will be used to compute (predict) the action for the cartpole [3].

$$Q(s,a) \approx Q(s,a,\overline{w})$$



Benchmark Model

Benchmark for this project will be based on one of the solutions given on OpenAI and evaluation the number of episode used before solving the game

Evaluation Metrics

In this project it was decided to follow the OpenAI specification and choose as a main metrics score of 195.0 which solve the cartpole game problem.

Project Design

The action of the cartpole will be predicted by trained model of convolutional neural network, which is going to be learned by used of DQN algorithm. Following the Bellman equation, the target of the game state can be defined as follows:

$$Q(s,a) = r + \gamma \cdot maxQ(s',a')$$

Where **s** is the environment state, a is an agent action, **s'** is the next state from state **s** and action **a**. **r** represents the maximum discounted future reward when we perform action **a** in state **s**. γ – discount factor.
In order to minimalize the distance between the predicted and target state the loss function has to be defined (below). Further, given function will be used to update the weights of neural network.

$$loss = \left(r + \gamma \cdot maxQ(s',a') - Q(s,a)\right)^2$$

In order to stabilized the learning process the weight of neural network will be updated by sampling a random batch of memory tuples at each timestep from memory. This will mainly reduce the variance and allow to learn from multiple samples not just from the recent transition.
The main strategy in this project is to approach the final solution by use of iterative process.
The skeleton project solution can be given as follows:

1. Elaboration of convolution network architecture (the architecture can be optimized during the iterative approach design process)
2. Approach the value of hyperparameters both neural network and reinforcement learning process (environment and algorithm).
3. Perform the training:
    a. Initialize the memory and neural network weights
    b. For *number_of_episodes* do:
        i. With probability ε (decaying exploration factor) chose the random action – in order to explore environment otherwise choose the action computing the $a_t = \arg maxQ(s,a)$
        ii. Agent of the game to execute the action a_t, receives (Q_decision) the reward and carpole goes to state s_{t+1}
        iii. Above transition will be stored in defined previously memory
        iv. Sample the memory to build the batch and perform the neural network training
        v. Current output from neural network is used to compute a new target (Bellman equation) or the game is over with the current reward
        vi. Compute the loss and propagate backward to update the weights
4. Save the solution (architecture and trained weights).
5. Test solution. Check if cartpole fulfills the project requirements (keep the pole upright for 195 frames).

References:

[1]. DeepMind technologies, Playing Atari with Deep Reinforcement Learning, https://www.cs.toronto.edu/~vmnih/docs/dqn.pdf

[2]. Tambet Matiiasen, Demystifying Deep Reinforcement Learning, https://www.intelnervana.com/demystifying-deep-reinforcement-learning/

[3]. David Silver, Deep Reinforcement Learning http://www0.cs.ucl.ac.uk/staff/d.silver/web/Resources_files/deep_rl.pdf