Task Planning in “Block World” with Deep Reinforcement Learning

Edward Ayunts, Alekasndr I. Panov

Abstract At the moment reinforcement learning have advanced significantly with discovering new techniques and instruments for training. This paper is devoted to the application convolutional and recurrent neural networks in the task of planning with reinforcement learning problem. The aim of the work is to check whether the neural networks are fit for this problem. During the experiments in a block environment the task was to move blocks to obtain the final arrangement which was the target. Significant part of the problem is connected with the determining on the reward function and how the results are depending in reward’s calculation. The current results show that without modifying the initial problem into more straightforward ones neural networks didn’t demonstrate stable learning process. In the paper a modified reward function with sub-targets and euclidian reward calculation was used for more precise reward determination. Results have shown that none of the tested architectures were not able to achieve goal

1 Introduction

In the robotics it is usually assumed the concept that robot is able to perform only the actions that it is programmed for. It enables them to do their tasks well, but limit their capacity to perform new actions. That is why the concept of learning robot is of greater interest, because in will potentially let it perform new complex actions,

Edward Ayunts
National Research University Higher School of Economics, Moscow, Russia
e-mail: eayunts@gmail.com

Aleksandr I. Panov
Federal Research Center “Computer Science and Control” of Russian Academy of Sciences, Moscow, Russia
National Research University Higher School of Economics, Moscow, Russia
e-mail: pan@isa.ru

using a few basic ones. The key part in developing such robot is its training, and the novice of this paper that authors use neural networks with various architectures for training the robot to perform complex multi-step action.

The work in this area was carried out for a last two decades and especially intensified in recent years. It's worth mention the research [1], in which humanoid robot was taught to play air-hockey using visual processing of the game-field, sub-targets and primitives - simple actions, that "can be combined to complete the task".

In a last few years neural networks became the catalysts for rapid and widespread popularity for the area. In the one of the key papers [2] in this field author managed to train agent to play Atari using Deep reinforcement learning. More precisely, agent received image at every step, preprocess it using 2D convolutions. The Q-Learning was modelled as neural network with observation as input and vector of separate units for each possible actions as output. The results show that the model was able to perform in some games better than the human.

In the [3] authors developed a model which lets robot to foresight consequences of its actions, analyse them and determine the most preferable action. Robot gets as input video frames, and what is important - not necessarily from the same point of view. After model training, the agent is able to move object in new environments. The authors claim that the agent will able to perform more complex actions if the more detailed environment picture will be processed for training. The novice of the work is that authors did not use a special environment. Instead, they feed the net of a couple LSTM layers with photographs, and predict the new pixels by maximal likelihood method. The agent copes with the problem of determining the centre of masses of the object and is able to perform rotation. The only problem is that agent cannot distinguish its manipulator from the objects.

In another paper [4] the main breakthrough is the training of the agent to reveal the parts of the picture worth paying attention and process them in high resolution. It gives a big advantage in using convolutional nets as there is no need to use them for processing the whole image, so it allows to economy on computations. The idea is adopted from human vision - the agent as a human, is focused only on a point from the whole picture. LSTM and Dense layers with ReLU were used for training the model.

In [5] the author formulated the problem as following: agent should by visual information rotate by fingers an object in hand. The uniqueness of the work is that there are no assumptions on configurations of the hand, or its physical parameters, only pixel data is used. The aim was to provide rotating in only one axis, such that the rotating in all other axises was minimal. The convolutional and fully connected layers were trained and eventually the target was designed so that provide movement only in vertical axis. After 11 mln iterations stable performance was obtained.

Value Iteration Networks, introduced in the [4], et al, are of great interest, as they let a neural network not just train in the learning process, but to plan agent's track. The algorithm "learns an explicit planning computations", which enables it to generalize better to new domains. In the article authors compare their model with the fully-connected and convolutional networks and show the training results in three different environments and problems, during which VIN shows stable higher performance, especially when the volume of data increases.

2 Problem Statement

The concept used for solution the stated problem is called Q-learning, which idea is the modelling the future cumulative reward for each action at every step. Initially, a zero matrix of dimension nS x nA is created, where nS and nA are the numbers of states and actions respectfully. At each step the agent chooses the action, such that the expected reward from it is maximal among all actions. After each step, the value of the expected reward is modifying by multiplying initial one by coefficient γ, usually equal to 0.99, and adding to it the received reward. The process of such iterations is called Q-learning. The key formula is $Q(s,a) = E(R|s,a,\pi)$, where Q is expected total payoff for choosing action a in the state s and if in the future, the same strategy will be followed.

At every step the future payoff is calculated as $R_t = \sum_{t'=t}^{T} \gamma^{t'-1} r_{t'}$, where r is the reward for the current step. The decision function for the agent is

$$Q*(s_t,a_t) = E_{s_t}[r_t + \gamma max].$$

Drawback of the algorithm is that there are situations in which proper action is depending on the environment, but the Q-learning does not take into account this point. That is why the hypothesis of the research is that using neural networks which take as input the whole vector of current state and of the target, the problem can be solved. Under the consideration three types of neural networks: multilayer perceptron, recurrent and convolutional networks.

3 Experiments

For conducting experiments the environment based on the OpenAI Gym was developed, available in `https://github.com/cog-isa/deep-blocks.git`. It consists of 30x30 matrix which illustrates the input signal on visual sensor, front-side view on the environment. There are 11 3x3 blocks in the environment, marked as ’1’, all other elements are zeros. The agent also gets target matrix, which also contains the same 11 blocks, but all collected in the centre. The agent has 8 actions - by 4 for movements with cubes and without them. The manipulator is also represented by ’1’s as reversed letter T in the 3x3 cube. The robot is able to move cube only when the manipulator is right above the cube. All restrictions, concerning physics of the movements, were imposed. Also every episode is limited to 100 moves. If the robot is aiming to perform prohibited movements, the state does not change and agent gets zero reward. Otherwise, reward is calculated as element-wise product of vectors of current observations and target divided by the total number of cubes. Multilayer perceptron with such reward-function demonstrated low performance, it hadn't succeed in achieving target, but at least, it was able to move almost all cubes towards their final locations, the problems arise with the cubes at corners . Also agent often get locked in the upper areas of the environment where are no cubes, so, such reward function was not informative. The experiments were also conducted using ordinary Q-learning algorithm.

4 Reward function

Instead of calculating it as proportion of blocks which are at their correct location at the current state, the euclidian distance is calculated between the manipulator and the block, which should be moved to its final place at current stage (their priorities are taken as given from initially given sub-targets). If the the manipulator is already has took the block, then the reward is the distance to block's final location. Reward is normed by dividing by $30\sqrt(2)$, to be less than 1. As soon as the block is at its place, the agent starts to calculate reward with use of the next subtarget. As can be seen from the algorithm below, the initial task is divided into 4 more simple ones. There 4 sub-targets because in the target blocks arrangement only 4 blocks are to be moved. So for each block to be moved a subtarget is designed. The reward is designed in such way to encourage agent to carry the block and don't do prohibited movements.

```
Data: Current State and Subtarget / Target
Result: Reward
Done1, Done2, Done3, Done4 = False, False, Fasle, False;
dones = [Done1, Done2, Done3, Done4];
For i in dones:;
if i is False then
    if The manipulator is not above the target block then
        Return 10 * ( 1- distance from manipulator to target block in current
          subtarget);
    else
        Return 20 * (1- distance from target block in current subtarget to its
          final location);
    end
else
end
```

Algorithm 1: Reward Calculation

5 Implemented Neural Architectures

The architecture of the used network consisted of 1800x1 input vector, two hidden layers with 1000 and 100 layers respectfully and a output layer of 8x1 dimension. Input is constructed from 900 element vector for current state and 900 for target, states are represented as reshaped from 30x30 to 900x1 vectors.

5.1 *Convolutional network*

The architecture for convolutional network was the following: 1D Convolution layer with 14 filter size and kernel = 1 was followed by two Dense layers with Dropout between them, and output dense layer which returns (1, 8) vector.

The results show that at current stage the performance of model is very low: agent didn’t manage to achieve at least first subtarget.

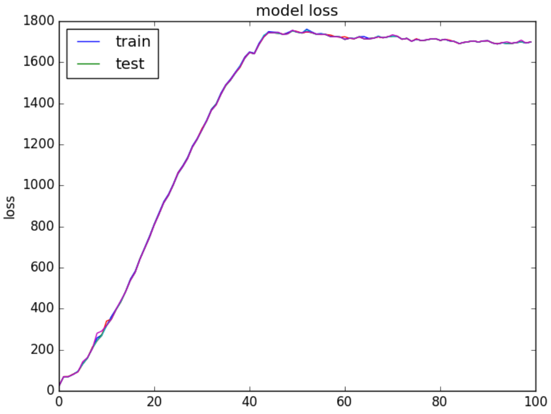

Fig. 1 A model’s loss trained on 100 episodes, replayed after each 50 steps.

5.2 *Recurrent net*

After unsatisfactory results with convolution the agent was trained on recurrent network with one LSTM layer. The architecture consisted of input vector with dimension (1,1800), LSTM with 1000 output elements, 1000 elements output Dense, 25% Dropout, Dense with 100 output elements and final layer with eight elements for each action.

The agent was trained on 10 000 episodes and the results are very unstable, and it didn't achieved goal. The same with two LSTM layers (second was added just after the first one). It was trained on the less number of episodes, and although it shows the more apparent downward trend, it also has not won the game.

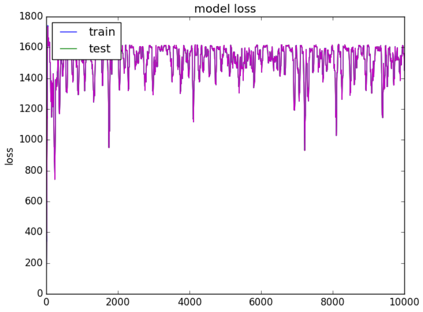

Fig. 2 Model with 1 LSTM layer

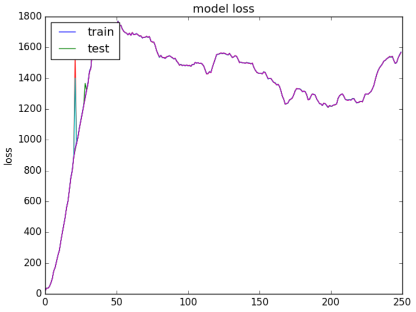

Fig. 3 Model with 2 LSTM layers

5.3 *Q-learning*

Using regular Q-learning table without neural network didn't gave more encouraging results. Using the initial reward function (fraction of cubes in their initial point) the mean reward was stable and equal roughly 0.5, it is less than initially and it means that the algorithm is less performative the Deep Q network. The reason for such result is that at every state the optimal movement depends in the context and Q-learning table does not consider other cubes location.

6 Testing the hypothesis with simple model

As was stated before, used approach didn't showed adoptable performance. In aim to check whether is it operates at all, the simple version of the game was designed. It consisted only of one block in the left corner of the 30x30 field and agent's task was to move it to the right corner. The same reward function was used, with the following adjustments:

1. Reward is equal to Stage Coefficient multiplied by 1 – Distance to target;
2. Stage Coefficient is equal 20 if block is under the manipulator and 10 otherwise;
3. Reward is equal zero if the state has not changed since the last action.

Conducted experiments using LSTM and 1D Convolution together showed that agent is able to move the block from left corner to the right one. It was made in 150 steps since the start of training. This result proves the initial hypothesis, that proposed approach is able to solve the task.

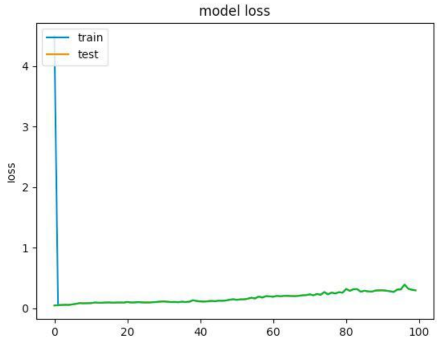

Fig. 4 Loss function of model which managed to perform task

7 Conclusions

In the designed environment agents was trained on Deep Q-Learning model implementation based on neural network with the used reward function proved to be unsuccessful for. Among all tested architectures the most promising was the one with two sequential LSTM and LSTM with 1D convolution layers. Nevertheless experiments with the same agent in the one block environment with the proposed approach showed good performance and so it has a potential on a more complex tasks. Conducted tests lead to a conclusion that the target would be attainable if more advanced reward policy will be used. The hypothesis of the causes of such results is that the model is underfitted because of relatively large size of the environment and plenty of unsuccessful actions which don’t provide information for the network. Further more advanced agent configurations and network architectures are planned to be tested and other planning tasks are to be worked on.

Acknowledgements The reported study was supported by RFBR, research Projects No. 16-37-60055 and No. 17-07-00281.

References

1. Darrin C. Bentivegna, Alles Ude, Christopher G. Atkenson, Gordon Cheng: Humanoid Robot Learning and Game Playing Using PC-Based Vision Switzerland, (2002). Darrin C. Bentivegna, Alles Ude, Christopher G. Atkenson, Gordon Cheng:
2. Volodymyr Mnih: Playing Atari with Deep Reinforcement Learning (NIPS 2013)
3. Chelsea Finn, Sergey Levine: Deep Visual Foresight for Planning Robot Motion (ICRA 2017).
4. Volodymyr Mnih, Nicolas Heess, Alex Graves, Koray Kavukcuoglu: Recurrent Models of Visual Attention (2014)
5. Kapil D. Katyal, Edward W. Staley, Matthew S. Johannes, I-Jeng Wang ,Austin Reiter, Phillipe Burlina : In-Hand Robotic Manipulation via Deep Reinforcement Learning (2017)