Proceedings of the International Conference on
Information Technologies (InfoTech-2018)
20-21 September 2018, Bulgaria

APPLICATION OF DEEP LEARNING APPROACH IN SEQUENTIAL GAMES

Vanya Markova, Ventseslav Shopov

Bulgarian Academy of Sciences – Institute of Robotics
e-mails: markovavanya@yahoo.com, vkshopov@yahoo.com
Bulgaria

Abstract: This article focuses on the application of deep learning techniques in sequential games. The main hypothesis is that sequence to sequence learning approach is applicable and demonstrate better performance than classic reinforcement learning. So autonomous agents trough sequence to sequence approach are capable of discovering good solutions to the problem at hand by learning in dynamic environment.
Key words: sequence to sequence, sequential games, autonomous agents

1. INTRODUCTION

The construction of an autonomous behavioral agent is an important issue with regard to the application of the results in various fields such as robotics, and autonomous decision-making. We study the relevance of the Sequence to Sequence(S2S) model for building autonomous agent plans in the Sequential Games (SG) area. In recent years, deep neural networks, and in particular the S2S approach, have been widely used in are of machine translation.

In this study we pose the question: is it S2S applicable to the approach of building agent plans in the field of Sequential Games. In addition, we study what is the impact of different optimization algorithms and attention mechanisms on learning performance of S2S approach. This research may enable us to better understand and control the autonomous agents behavior in complex Sequential games.

The reward function is an very import in regard of transferring the knowledge gain. Ng investigates conditions under which modifications to the reward function of a MDP preserve the optimal policy [10]. Konidaris introduces shaping rewards in

reinforcement learning tasks, that result in accelerated learning in late tasks that are related but distinct [9].

Game Theory is a well-established paradigm for shaping behavior and building up plans in the field of autonomous agents and systems. Models and algorithms in the field of machine learning provide theoretical solutions based on the learned behavior patterns [12, 13, 14]. Recently, using the power of deep learning modeling, strengthening learning has been successfully used for numerous tasks, including Atari and Go, robotic manipulations and sequential data generation [16].

Sequence to sequence neural networks are two encoder and decoder blocks, and some hidden inner state layer that connects them[6]. In turn, the encoder consists of a chain of recurrent cells (in implementation it can be either one or several)[7]. In recent years, several articles have been published that successfully apply S2S models in the field of machine translation [2,3]. There are several [4,5] solutions based on Recurrent Neural Networks RNN and Long Short Term Memory.

In this paper we study the applicability of the S2S approach to build plans in a dynamic environment. In addition, we experimentally investigate the impact of some parameters on the S2S model on the efficiency of learning. Our main hypothesis is that S2S are capable to build adequate sequences of action in sequential games.

The article is organized as follows: in the second part we give a brief description of the theoretical foundations and the implementation of a proposed solution. In the third part we describe the experimental setting and present the obtained results . In the final part we discuss the applicability of S2S in the field of building autonomous agents' behavior.

2. METHODS AND MATERIALS

By describing some of the important issues of building a behavior in a dynamic environment, the traditional framework of game theory can not represent the complete complexity of agent training. An important part of the problem is to make rational solutions in a state of transition. That is why we are now looking at an extended framework in which we generalize both sequential games and MDP.

2.1 Markov Decision Process

Markov Decision Process (MDP): We formulate the transfer learning problem in sequential decision making domains using the following framework of Markov Decision Process.

We use the following definition of MDP as a 5-tuple $< S, A, T, R, g >$ where the set of states, set of actions, transition function and reward function are described. And $P : S \times A \rightarrow T(S)$ (2) is a transition function that maps the probability of moving to a new state given an action and the current state, $R : S \times A \rightarrow R$ is a reward

function, that gives the immediate reward of taking an action in a state. And $g \in [0, 1)$ is the discount factor.

So, the MDP of the agent is described tuple where *S* is the set of states, *A* is the set of actions, *P* is transition function and *R* is a reward function. The transition function *P* maps the the probability of moving to a new state given an action and the current states. The reward functions *R* that gives the immediate reward of taking an action and the discount factor *g*.

2.2 Sequence to sequence neural networks

Sequence to sequence neural networks are two encoder and decoder blocks, and some hidden inner state layer that connects them. In turn, the encoder consists of a chain of recurrent cells (in implementation it can be either one or several).

Recurrent Neural Network (RNN) [16] is a natural generalization of neural networks to sequences. Given the sequence of inputs *(x1, ..., xT),* the standard RNN computes a sequence of outputs *(y1, ..., yT)* by iteration. RNN can easily assign sequences to sequences when alignment between inputs is previously known. However, it is not easy to implement RNN for problems where input and output sequences have different lengths.

One simple strategy for general sequence training is to translate the input sequence to a fixed-size vector using an RNN and then to transfer the vector in the target sequence to another RNN.

Even when all relevant information is provided to RNNs, it is difficult to train RNNs due to long-term dependencies. To overcome this difficulty, you need to apply a memory method. Long Short Term Memory (LSTM) [1] is an approach that can resolve these dependencies. The purpose of LSTM is to evaluate the probability p *(y1, ..., yTo | x1, ..., xTi)* where *(x1, ..., xTi)* is an input sequence and y1 ,. ,, yT ' is its corresponding output sequence whose length Ti may be different from To.

In addition to the basic S2S approach, we will look at two algorithms that apply the mechanism of caution, this approach was first introduced by Bahdanau [19], then refined by Luong [8]. The main idea of the mechanism of attention is to establish direct links between the target and the source, paying attention to the relationship between states and actions.

We also explore the impact of two optimization methods (AdaGrad and Adam), using the approach described in [18]. In AdaGrad, some parameters are updated much more frequently than others, and so the training frequency for each parameter is dynamically adjusted [15]. Adam is now a popular optimization method, as it greatly accelerates convergence and thus reduces experimental cycles. The deficiency of the method is, that it is prone to over-fitting [17].

2.3 Implementation

We represent the environment as a discreet *NxN* game as shown in Figure 1. The goal of the agent is to move from the starting position to the ultimate goal by avoiding obstacles and collecting prizes. In order for this task to be accomplished

with a minimum number of actions, the agent should use a rational approach to building a plan.

In our study, we will seek to translate the map of the environment into the sequence of actions, goals, and rewards. In this way, we are as close as possible to the main purpose of the S2S models: the translation of an incoming sequence from one "language" into a corresponding sequence from another "language".

For this purpose, we have to construct from the environment a sequence of symbols. We create such a sequence by encoding the starting position, end goal, obstacle, and rewards as symbols. Moreover, in the case of 2-dimensional "world" we simply "unfold" the two-dimensional map in one-dimensional sequence as shown in Figure 2.

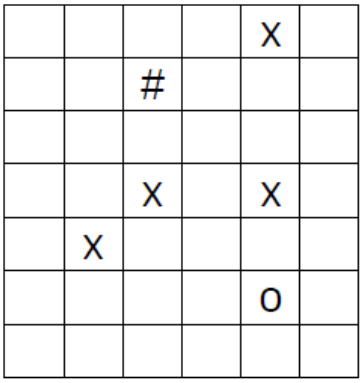

Fig. 1. The Environment is defined as a NxN grid

The problem is to give a sequence of actions from a set of actions ["left", "right", "up", "down", "take", "attack"] in such way that our agent could reach “destination point” in reasonable way as is shown on Figure 3.

Fig. 2. The input sequence that represents enviromnent

At the entrance, we need to submit the state of our world, consisting of separate cells, each of which can take one value from the set: ["space", "enemy", "life", "source point", "destination point"].

Fig. 3. The output sequence of actions

We can display such a world in the form of a vector of 6 * 25 dimensions, and after that compressing this representation trough embedded algorithm. How ever

such a model will be very sensitive to the change in the number of cells and objects in this world.

To get rid of this limitation, we can form the input layer as a sequence, where each element of this sequence is one object of our world. Thus, we will input sequences of different lengths (for different sizes of the simulated world) and in the process of training we will be able to expand the number of objects in our world.

3. EXPERIMENTS AND RESULTS

We present the experimental settings as follows:

First we "construct" the neural network using TensorFlow[] and Python[].

1. The first thing to do is define the input layers:

2. Next, create the encoder layer.

3. Here it is necessary to say that to reduce the dimension, the embedding mechanism is used, the mechanics of its implementation are already present in TensorFlow.

4. Create RNN cell and add them to our network.

5. The output of our subnet will consist of the output (conveyor) of the last RNN cell and its hidden state. We need only state.

6. We pass to the decoder.

7. Just like in the decoder, you need to prepare a layer of embedding.

8. Next, create the first layer with recurrent cells and project their outputs onto a fully connected perceptron for further classification of the results.

The outputs of the cells are fed to the fully connected layer of the classifier. In the decoder we will have two branches of the graph. The first branch is for learning, the other is for processing the final tasks. For training, we need to remove the last object from the target (those we want to get at the decoder output) of the sequences and add "GO" to the beginning of each target sequence. This is necessary, since we will train each cell separately and each of them must be given the correct input signal, and not the signal from the neighboring learning cell.

```
[1525113964.7153]  Decoder is ready.
[1525113964.7154]  Creating loss and optimization ...
[1525113966.2723]  Network is ready.
 Epoch 1/100      Batch 20/65     Loss: 1.161     Validation loss: 1.279     Time: 0.0129s
 Epoch 1/100      Batch 40/65     Loss: 1.129     Validation loss: 1.263     Time: 0.0120s
 Epoch 1/100      Batch 60/65     Loss: 1.169     Validation loss: 1.192     Time: 0.0131s
 Epoch 2/100      Batch 20/65     Loss: 0.987     Validation loss: 1.088     Time: 0.0158s
 Epoch 2/100      Batch 40/65     Loss: 0.902     Validation loss: 1.005     Time: 0.0128s
 Epoch 2/100      Batch 60/65     Loss: 0.851     Validation loss: 0.882     Time: 0.0134s
 Epoch 3/100      Batch 20/65     Loss: 0.698     Validation loss: 0.792     Time: 0.0164s
 Epoch 3/100      Batch 40/65     Loss: 0.670     Validation loss: 0.725     Time: 0.0131s
```

Fig. 4. The training is builded upon TensorFlow

In this study we will examine the convergence of the loss function as an assessment of the convergence of the learning process. In addition, we will examine the impact of the decoder type on the performance of the training scheme. We will explore a baseline approach and two Attention Mechanism cases:

- base S2S
- Attention Mechanism - Bahdanau
- Attention Mechanism - Luong.

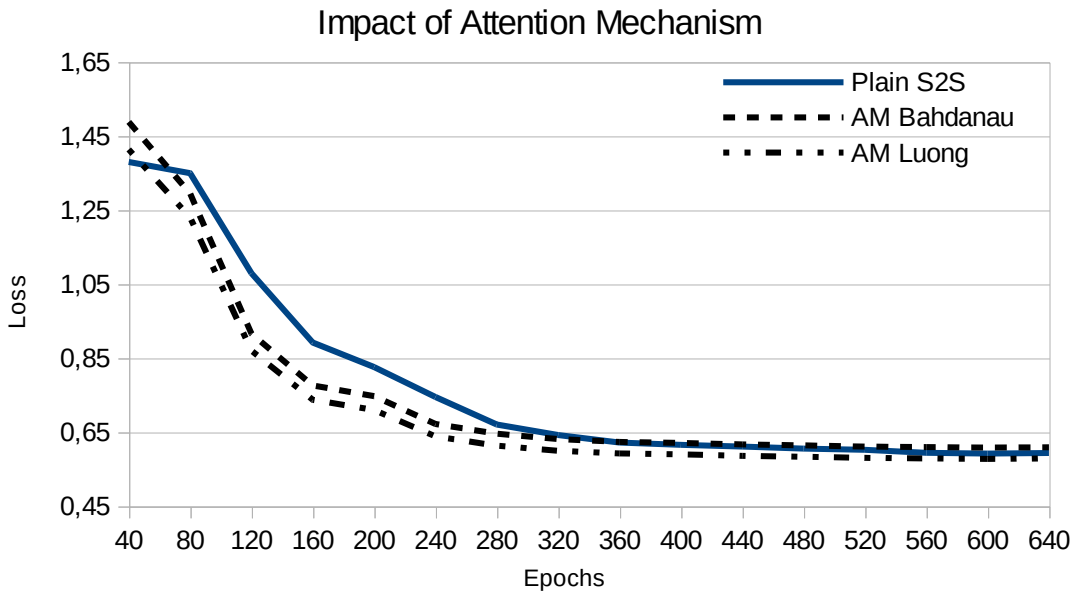

Fig. 5. The impact of Attention Mechanism on learning performance

In the second experiment we will investigate the impact of the optimization method on the encoder and decoder. We will compare the performance of the learning process for two approaches:

- AdaGrad
- Adam

For both experiments, five runs are made and Winzorized mean is taken, excluding 20% of the maximum and minimum elements. As a result, we get a sequence of steps to pass through our virtual labyrinth.

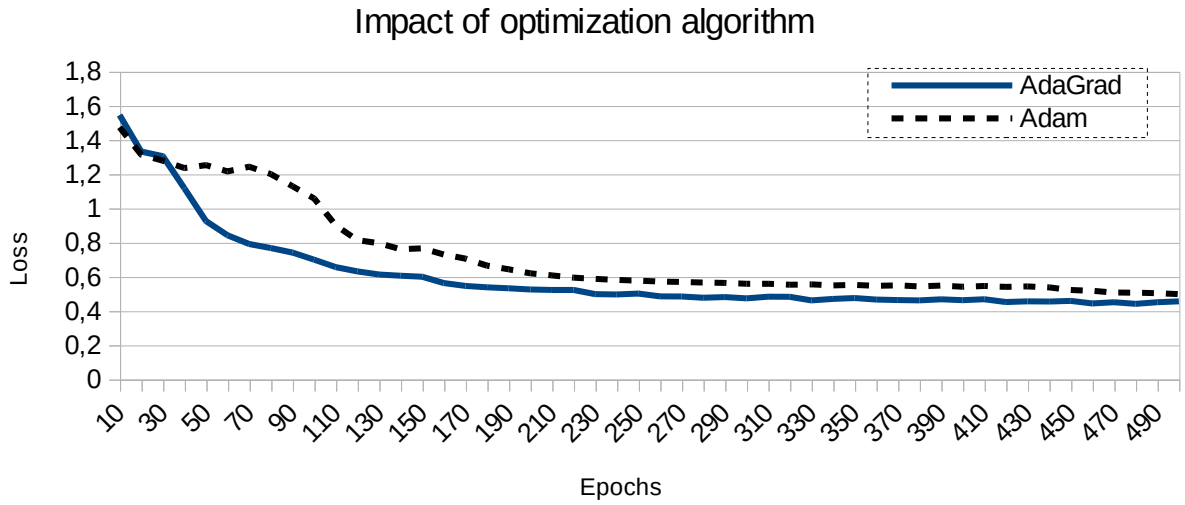

Fig. 6. The impact of optimization alghorithm

The summary reward is used as a measure of performance.

4. CONCLUSION

The results show that sequence to sequence learning approach is applicable and demonstrate better performance than classic reinforcement learning. So autonomous agents trough sequence to sequence approach are capable of discovering good solutions to the problem at hand by learning in dynamic environment.

The impact of different factors for building of agent behaviour in sequential games is discussed in this paper. The sequence to sequence approach is applied based on deep learning framework TensorFlow.

The summary reward is used as a measure of performance. The results show that sequence to sequence learning approach is applicable and demonstrate better performance than classic reinforcement learning. So autonomous agents trough sequence to sequence approach are capable of discovering good solutions to the problem at hand by learning in dynamic environment.

REFERENCES

[1] Hochreiter, S., & Schmidhuber, J. **(1997).** Long short-term memory. *Neural computation*, 9(8), 1735-1780.

[2] Yao, K., Cohn, T., Vylomova, K., Duh, K., & Dyer, C. **(2015)**. Depth-gated recurrent neural networks. *arXiv preprint.*

[3] Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R., & Schmidhuber, J. **(2017)**. LSTM: A search space odyssey. *IEEE transactions on neural networks and learning systems*, 28(10), 2222-2232.

[4] Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. **(2014)**. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078.*

[5] Jozefowicz, R., Zaremba, W., & Sutskever, I. **(2015, June)**. An empirical exploration of recurrent network architectures. *In International Conference on Machine Learning* (pp. 2342-2350).

[6]Kalchbrenner, N., & Blunsom, P. **(2013).** Recurrent continuous translation models. In Proceedings of the 2013 *Conference on Empirical Methods in Natural Language Processing* (pp. 1700-1709).

[7] Sutskever, I., Vinyals, O., & Le, Q. V. **(2014).** Sequence to sequence learning with neural networks. *In Advances in neural information processing systems* (pp. 3104-3112).

[8] Luong, M. T., Le, Q. V., Sutskever, I., Vinyals, O., & Kaiser, L. **(2015)**. Multi-task sequence to sequence learning. *arXiv preprint arXiv:1511.06114.*

[9] Konidaris, G., and Barto, A. **(2006)**. Autonomous shaping: Knowledge transfer in reinforcement learning. *In Proceedings of the 23rd international conference on Machine learning*, 489–496. ACM.

[10] Ng, A. Y.; Harada, D.; and Russell, S. **(1999)**. Policy invariance under reward transformations: Theory and application to reward shaping. *In ICML*, volume 99, 278–287.

[11]Cho, K., B. Merrienboer, C. Gulcehre, F. Bougares, H. Schwenk, and Y. Bengio. **(2014)** Learning phrase represen-tations using RNN encoder-decoder for statistical machinetranslation. *InArxiv preprint arXiv:*1406.1078,.

[12] Xu, H.; Ford, B.; Fang, F.; Dilkina, B.; Plumptre, A.; Tambe, M.; Driciru, M.; Wanyama, F.; Rwetsiba, A.; Nsubaga, M.; et al. **(2017)** Optimal patrol planning for green security games with black-box attackers. *In International Conference on Decision and Game Theory for Security*, 458–477.Springer.

[13] Kar, D.; Ford, B.; Gholami, S.; Fang, F.; Plumptre, A.; Tambe, M.; Driciru, M.; Wanyama, F.; Rwetsiba, A.; Nsubaga, M.; et al. **(2017).** Cloudy with a chance of poaching: Adversary behavior modeling and forecasting with real-world poaching data. In Pro*ceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems* , 159–167.

[14] Gholami, S.; Ford, B.; Fang, F.; Plumptre, A.; Tambe, M.; Driciru, M.; Wanyama, F.; Rwetsiba, A.; Nsubaga, M.; and Mabonga, J. **(2017)** Taking it for a test drive: a hybrid spatio-temporal model for wildlife poaching prediction evaluated through a controlled field test. *In Proceedings of the European Conference on Machine Learning & Principles and Practice of Knowledge Discovery in Databases, ECML PKDD*

[15] Duchi, J., Elad Hazan, and Yoram Singer. **(2011)** Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research, 12:2121–2159*

[16] Sutskever, I., Vinyals, O., & Le, Q. V. **(2014)**. Sequence to sequence learning with neural networks. *In Advances in neural information processing systems (pp. 3104-3112).*

[17] Diederik Kingma and Jimmy Ba. **(2014)** Adam: A method for stochastic optimization. *ArXiv preprint arXiv*:1412.6980,.

[18] Neubig, G. **(2017).** Neural machine translation and sequence-to-sequence models: A tutorial. *arXiv preprint arXiv:1703.01619*.

[19] Bahdanau, D., Cho, K., & Bengio, Y. **(2014)**. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv*:1409.0473.