DEEP NEURAL NETWORKS IN COLLECTIVE ROBOTICS CONTROL

Vanya Markova[1]
markovavanya@yahoo.com

Ventseslav Shopov[1]
vkshopov@yahoo.com

[1]Institute of Robotics
Bulgarian Academy of Sciences

August 10, 2020

Table of Contents

1. Introduction
   - The Basic Problem That Is Studied
2. Methods and Materials
   - Reinforcement Learning
   - Recurrent Neural Networks for Optimization Problems
   - Recurrent Neural Ntetworks
     - Long Short Trem Memory
     - Peephole LSTM
3. Experiments and Results
4. Conclusion
5. Refernces
6. Referneces

Outline

1. Introduction
   - The Basic Problem That Is Studied
2. Methods and Materials
   - Reinforcement Learning
   - Recurrent Neural Networks for Optimization Problems
   - Recurrent Neural Ntetworks
     - Long Short Trem Memory
     - Peephole LSTM
3. Experiments and Results
4. Conclusion
5. Refernces
6. Refernces

The topic of the study

is the research on the coordination of many robotic systems. This is largely due to many practical applications such as

- urban search and rescue [1],
- surveillance and intelligence [2],
- environmental monitoring [3],
- and mapping of unknown environments [4].

Main goal

Generally, the problem with the formation of many robotic systems involves two steps:

- first determining the desired formation, which is beyond the scope of this study
- and on the second hand designing the appropriate control algorithm to reach and maintain this formation.

Outline

1. Introduction
   - The Basic Problem That Is Studied
2. Methods and Materials
   - Reinforcement Learning
   - Recurrent Neural Networks for Optimization Problems
   - Recurrent Neural Ntetworks
     - Long Short Trem Memory
     - Peephole LSTM
3. Experiments and Results
4. Conclusion
5. Refernces
6. Refernces

Reinforcement Learning

In general, the reinforcement learning process used is a cycle that begins with the observation of a given state s_t, selects an action a_t, waits for the action to complete, observes the received state s_{t+1}, and accordingly updates the evaluation of its value function and the next cycle begins.
Since updating the value estimate of a feature is the most often expected intensive step, we have introduced the idea of an update buffer. It caches the observations of s_{t+1} and the reward and delays the evaluation of the value function until the next time the robot waits for the action of its choice.

Recurrent Neural Networks for Optimization Problems

The key moment for building an optimal formation is to effectively solve the optimization problem. Unlike conventional numerical methods, the efficiency of recurrent neural network does not decrease as the size of the optimization problem [7] [8], [11].
One common assumption of the aforementioned neural networks is that the optimization problem is smooth.In [13], a generalized neural network based on the model proposed in [12] was presented for non-smooth problems with nonlinear programming. Extensions of the non-smooth convex optimization problem are made in [14].

Lorenz attractor

Numerical methods are often used to analyze the structure of the attractor

- For example, a combination of an explicit Euler method with a central difference scheme.
- Other methods are the Adams method using higher derivatives
- Runge-Kutta methods

The equilibrium positions of the system are of saddle type. As a sequence this limits the use of these methods. Becouse small changes in the initial conditions of the system lead to significant consequences over time.

Recurrent Neural Ntetworks

Basic RNNs are a network of neuron-like nodes organized into successive "layers", each node in a given layer is connected with a directed (one-way) connection to every other node in the next successive layer.

Each node (neuron) has a time-varying real-valued activation. Each connection (synapse) has a modifiable real-valued weight. Nodes are either input nodes (receiving data from outside the network), output nodes (yielding results), or hidden nodes (that modify the data en route from input to output).

Long Short Trem Memory

There are several architectures of LSTM units. A common architecture is composed of a memory cell, an input gate, an output gate and a forget gate. This is equivalent to applying the identity functioni $f(x) = x$ to the input.

Because the derivative of the identity function is constant, when an LSTM network is trained with backpropagation through time, the gradient does not vanish. The activation function of the LSTM gates is often the logistic function. There are connections into and out of the LSTM gates, a few of which are recurrent. The weights of these connections, which need to be learned during training, determine how the gates operate.

LSTM with a forget gate

In the equations below, the lowercase variables represent vectors. Matrices W_q and U_q contain, respectively, the weights of the input and recurrent connections, where q can either be the input gate i, output gate o, the forget gate f or the memory cell c c, depending on the activation being calculated. The compact forms of the equations for the forward pass of an LSTM unit with a forget gate are:

$$
\begin{aligned}
f_t &= \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \\
i_t &= \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \\
o_t &= \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \\
c_t &= f_t \circ c_{t-1} + i_t \circ \sigma_c(W_c x_t + U_c h_{t-1} + b_c) \\
h_t &= o_t \circ \sigma_h(c_t)
\end{aligned}
$$

LSTM with a forget gate

The initial values are $c_0 = 0$ and $h_0 = 0$ and the operator $\circ$ denotes the Hadamard product (element-wise product). The subscript t indexes the time step.

Variables:

- $x_t \in R^d$ is input vector to the LSTM unit;
- $f_t \in R^h$ is forget gate's activation vector;
- $i_t \in R^h$ is input gate's activation vector;
- $o_t \in R^h$ is output gate's activation vector;
- $h_t \in R^h$ is output vector of the LSTM unit;
- $c_t \in R^h$ is cell state vector;

$W \in R^{h \times d}$, $U \in R^{h \times h}$ and $b \in R^h$ are weight matrices and bias vector parameters which need to be learned during training where the superscripts d and h refer to the number of input features and number of hidden units, respectively.

Peephole LSTM

The figure on the right is a graphical representation of an LSTM unit with peephole connections (i.e. a peephole LSTM). Peephole connections allow the gates to access the constant error carousel (CEC), whose activation is the cell state. h_{t-1} is not used, c_{t-1} is used instead in most places.

$$
\begin{aligned}
f_t &= \sigma_g(W_f x_t + U_f c_{t-1} + b_f) \\
i_t &= \sigma_g(W_i x_t + U_i c_{t-1} + b_i) \\
o_t &= \sigma_g(W_o x_t + U_o c_{t-1} + b_o) \\
c_t &= f_t \circ c_{t-1} + i_t \circ \sigma_c(W_c x_t + U_c c_{t-1} + b_c) \\
h_t &= o_t \circ \sigma_h(c_t)
\end{aligned}
$$

Outline

1. Introduction
   - The Basic Problem That Is Studied
2. Methods and Materials
   - Reinforcement Learning
   - Recurrent Neural Networks for Optimization Problems
   - Recurrent Neural Ntetworks
     - Long Short Trem Memory
     - Peephole LSTM
3. Experiments and Results
4. Conclusion
5. Refernces
6. Refernces

Experiments

We have a group of robots moving together in formation. The formation is led by a leader, with the rest of the robots orienting themselves to their position in the leader formation. The problem of the leader's dismissal is solved by taking his place from another robot. All robots have an arrangement to take the leader position.

We have N robots and N positions that the robots have to position themselves. When we have heterogeneous robots, (ie, each robot has a specific position), the solution to the problem is trivial.

Comparison of Naive Approach and the Deep Learning

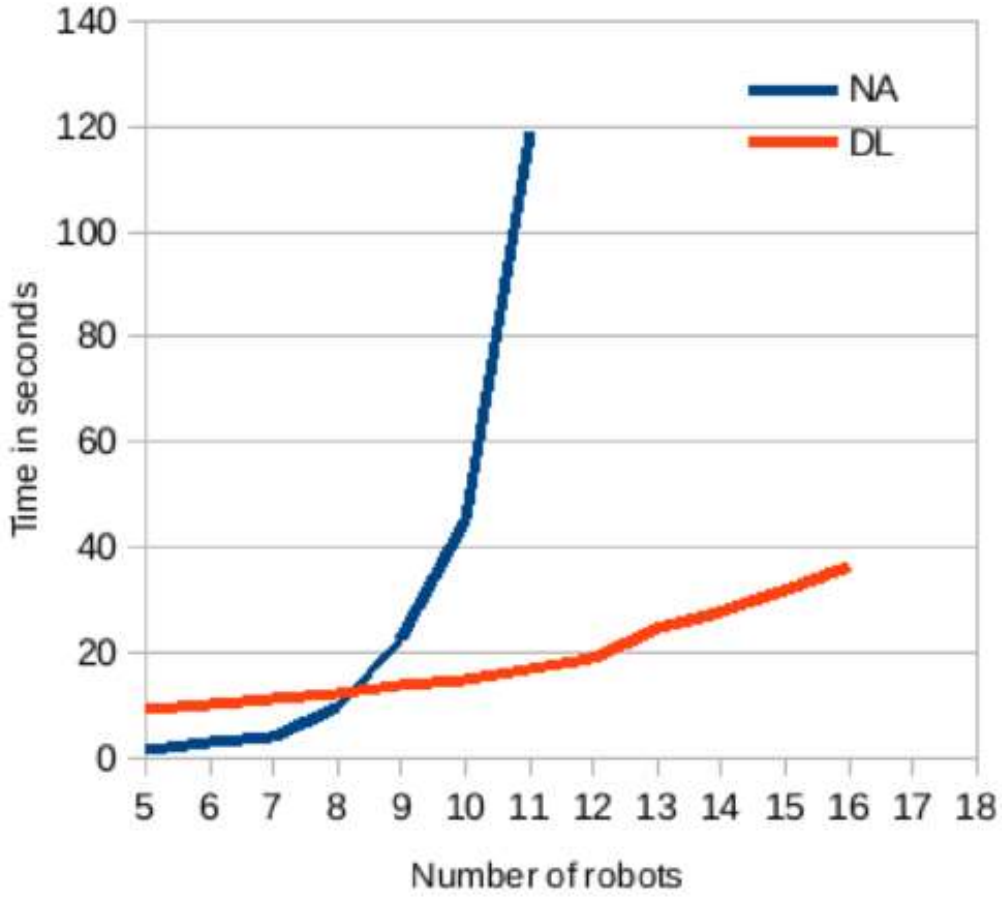

Figure: Comparison of Naive Approach and the Deep Learning

Results

The results showed that Deep Learning approach performs better naive approach.

Examines impact of training data set size on the accuracy of prediction

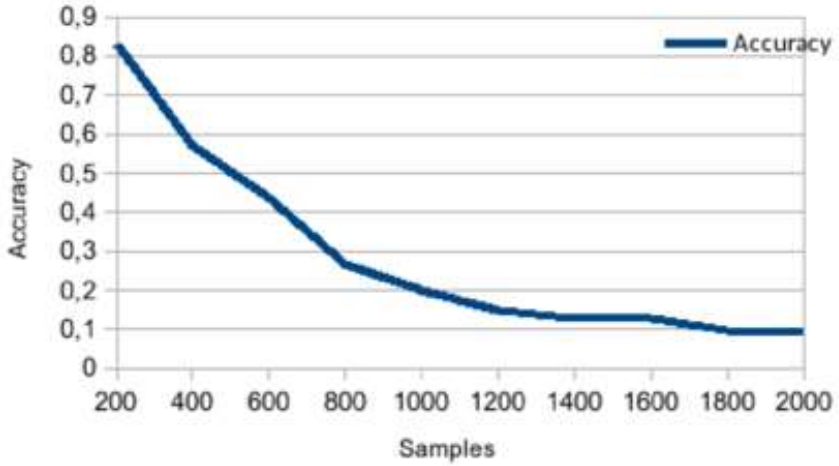

Figure: impact of training data set size on the accuracy of prediction

Results

Both experiments take five runs and the Winzorized mean is calculated.

For a larger number of robots in certain formation configurations, the last occupied position can be expected to be occupied much later than others. To avoid this situation, the robots should distribute their positions so that the total path to occupying these positions is minimal.

Outline

1. Introduction
   - The Basic Problem That Is Studied
2. Methods and Materials
   - Reinforcement Learning
   - Recurrent Neural Networks for Optimization Problems
   - Recurrent Neural Ntetworks
     - Long Short Trem Memory
     - Peephole LSTM
3. Experiments and Results
4. Conclusion
5. Refernces
6. Refernces

Conclusion

In this work, we propose a method for applying the training algorithm to assist the formation of groups of agents in a leader scenario.

Mobile agent formation control is an important issue for the collective of robots. Especially when they move independently without human oversight, controlling the movement and forming a collective of agents is a critical and challenging task.

To exercise control through supportive learning, we present the problem of formation as a Markov decision-making process. This allows us to use deep reinforcement learning to obtain the law of leadership of the successor leader.

Outline

1. Introduction
   - The Basic Problem That Is Studied
2. Methods and Materials
   - Reinforcement Learning
   - Recurrent Neural Networks for Optimization Problems
   - Recurrent Neural Ntetworks
     - Long Short Trem Memory
     - Peephole LSTM
3. Experiments and Results
4. Conclusion
5. Refernces
6. Refernces

Refernces

[1] Casper, J., & Murphy, R. R. (2003). Human-robot interactions during the robot-assisted urban search and rescue response at the world trade center. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 33(3), pp. 367-385.

[2] Saptharishi, M., Oliver, C. S., Diehl, C. P., Bhat, K. S., Dolan, J. M., Trebi-Ollennu, A., & Khosla, P. K. (2002). Distributed surveillance and reconnaissance using multiple autonomous ATVs: CyberScout. IEEE Transactions on Robotics and Automation, 18(5), pp. 826-836.

[3] Paley, D. A., Zhang, F., & Leonard, N. E. (2008). Cooperative control for ocean sampling: The glider coordinated control system. IEEE Transactions on Control Systems Technology, 16(4), pp. 735-744.

[4] Zhu, A., & Yang, S. X. (2007). Neurofuzzy-based approach to mobile robot navigation in unknown environments. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 37(4), pp. 610-621.

[5] Spletzer, J. R., & Fierro, R. (2005, April). Optimal positioning strategies for shape changes in robot teams. In Proceedings of the 2005 IEEE International Conference on Robotics and Automation pp. 742-747.

[6] Derenick, J. C., & Spletzer, J. R. (2007). Convex optimization strategies for coordinating large-scale robot formations. IEEE Transactions on Robotics, 23(6), pp. 1252-1259.

[7] Xia, Y., & Wang, J. (1998). A general methodology for designing globally convergent optimization neural networks. IEEE Transactions on Neural Networks, 9(6), pp. 1331-1343.

[8] Kamel, M. S., & Xia, Y. (2009). Cooperative recurrent modular neural networks for constrained optimization: a survey of models and applications. Cognitive neurodynamics, 3(1), pp. 47-81.

[9] Fei, Y., Rong, G., Wang, B., & Wang, W. (2014). Parallel L-BFGS-B algorithm on gpu. Computers & graphics, VOL. 40, pp. 1-9.

[10] Ploskas, N., & Samaras, N. (2015). Efficient GPU-based implementations of simplex type algorithms. Applied Mathematics and Computation, 250, pp. 552-570.

[11] Wang, J., & Kusiak, A. (2000). Neural network applications in intelligent manufacturing: An updated survey. In Computational intelligence in manufacturing handbook pp. 45-72.

[12] Kennedy, M. P., & Chua, L. O. (1988). Neural networks for nonlinear programming. IEEE Transactions on Circuits and Systems, 35(5), pp. 554-562.

[13] Forti, M., Nistri, P., & Quincampoix, M. (2004). Generalized neural network for nonsmooth nonlinear programming problems. IEEE Transactions on Circuits and Systems I: Regular Papers, 51(9), pp. 1741-1754.

[14] Xue, X., & Bian, W. (2008). Subgradient-based neural networks for nonsmooth convex optimization problems. IEEE Transactions on Circuits and Systems I: Regular Papers, 55(8), pp. 2378-2391.