ESANN 2014 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 23-25 April 2014, i6doc.com publ., ISBN 978-287419095-7.
Available from http://www.i6doc.com/fr/livre/?GCOI=28001100432440.

An application of the temporal difference algorithm to the truck backer-upper problem

Christopher J. Gatti and Mark J. Embrechts

Rensselaer Polytechnic Institute
Dept. of Industrial and Systems Engineering
Troy, NY - USA

Abstract. We use a reinforcement learning approach to learn a real world control problem, the truck backer-upper problem. In this problem, a tractor trailer truck must be backed into a loading dock from an arbitrary location and orientation. Our approach uses the temporal difference algorithm using a neural network as the value function approximator. The novelty of this work is the simplicity of our implementation, yet it is able to successfully back the truck into the loading dock from random initial locations and orientations.

1 Introduction

Reinforcement learning is an approach to solving sequential decision making problems that is based on the process of trial and error learning. This type of learning has been used in various types of benchmark control problems, including the mountain car problem [5] and the pole swing-up task [1]. However, aside from a few notable examples (i.e., helicopter control [6]), its application in real world control problems is rather limited.

In this work, we apply reinforcement learning to the truck backer-upper (TBU) problem. In this problem, a tractor trailer truck must be backed into a loading dock by controlling the orientation of the wheels of the truck cab. This problem has been considered in other works, albeit using slightly different or more complex approaches. Nguyen and Widrow [7, 8] used a neural network-based self-learning control system approach to back up a single trailer truck, though this was not based on a reinforcement learning approach. Vollbrecht [11] used a complex hierarchical reinforcement learning approach based state space partitioning using a kd-trie and a tabular Q-function to learn how to back up a single trailer truck. Our work is novel because we use a straight-forward and simplistic reinforcement learning approach to learn the truck backer-upper problems. More specifically, we use TD(λ) and a neural network for the function approximator to successfully learn this domain. Note that the purpose of this work was not to try to beat the state of the art for the TBU problem, but rather to determine if a relatively simple approach could be used.

2 Truck-backer upper problem

The goal of the truck backer-upper problem is to learn a control system that is capable of backing up a tractor trailer truck from an arbitrary location and

ESANN 2014 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 23-25 April 2014, i6doc.com publ., ISBN 978-287419095-7.
Available from http://www.i6doc.com/fr/livre/?GCOI=28001100432440.

orientation/configuration to a loading dock (Figure 1).

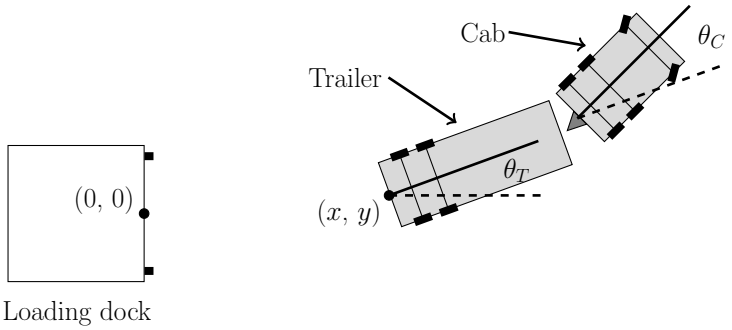

Fig. 1: The state of the truck is defined by the rear trailer position (x,y), the trailer angle θ_T, and the cab angle θ_C. The goal of the problem is to back the truck into the loading dock at $(x, y) = (0, 0)$ where $\theta_T = 0$.

The dynamics of the trailer truck were based on those from [9]. The position of the rear of the trailer was defined by its horizontal and vertical coordinates, x and y (meters), respectively. The orientation of the trailer with respect to a horizontal axis was defined by θ_T, and the orientation of the cab with respect to the trailer was defined by θ_C (radians). The state of the trailer s_t at any time step t was characterized by these four state variables: $s_t = (x, y, \theta_T, \theta_C)$. The state update equations are as follows:

$$
\begin{aligned}
x' &= x - B \cdot cos(\theta_T) \\
y' &= y - B \cdot sin(\theta_T) \\
\theta'_T &= \theta_T - arcsin\left(\frac{A \cdot sin(\theta_C)}{L_T}\right) \\
\theta'_C &= \theta_C + arcsin\left(\frac{v \cdot sin(u)}{L_C + L_T}\right)
\end{aligned}
$$

where $A = v \cdot cos(u)$, $B = A \cdot cos(\theta_C)$, $v = 3$, $L_T = 14$ (tailer length), and $L_C = 6$ (cab length). The wheel angle relative to the cab angle is specified by u (radians), and three discrete actions were allowed: $u = \{-1, 0, 1\}$. The truck velocity was not taken into account as backing the trailer is assumed to be a slow process. The truck was restricted to the domain boundaries $x = [0, 200]$ and $y = [-100, 100]$. The goal of this problem was to have the trailer positioned at the loading dock with a specific orientation: $x = 0$, $y = 0$, and $\theta_T = 0$.

3 Reinforcement learning implementation

The temporal difference algorithm TD(λ) [10] was used to train a neural network to learn the value function $V(s_t, a_t)$ that approximates the value of being in state s_t and taking action a_t at time t. The work described herein assumes some

ESANN 2014 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 23-25 April 2014, i6doc.com publ., ISBN 978-287419095-7.
Available from http://www.i6doc.com/fr/livre/?GCOI=28001100432440.

knowledge about reinforcement learning using a neural network, and the reader is directed to [2, 3] for additional background information.

3.1 Neural network

The neural network is used to evaluate the state value function $V(s_t, a_t)$ by propagating the current state $s_t = [x, y, \theta_T, \theta_C]$ through the network. The primary advantage of using a neural network is its ability to generalize to unvisited states, and this becomes essential in large or continuous state spaces. This network used four input nodes (for the four state variables), 51 hidden nodes, and three output nodes, which correspond to the three available actions. The x and y components of the state vector were scaled over $[-3, 3]$ based on the boundaries of the domain in order to put these state values on approximately the same scale as θ_T and θ_C. The hidden layer used a hyperbolic tangent (tanh) transfer function and the output layer used a linear transfer function. Network weights were initialized by sampling from $U[-0.1, 0.1]$.

The learning rates α of the network were set individually by layer following the approach described in [4]. Input-hidden (α_{hi}) and hidden-output (α_{oh}) learning rates are initially set to $1/\sqrt{n}$ where n is the number of nodes in the preceding layer, and α_{oh} is then divided by $\sqrt{3}$. All learning rates are divided by $\frac{1/(4 \cdot \phi)}{min(\alpha)}$, where $\phi = 500$ is a scale factor that is problem dependent and is related to the maximum number of time steps allowed. The resulting learning rates were $\alpha_{hi} \approx 0.0031$ and $\alpha_{oh} = 0.0005$.

3.2 TD(λ)

TD(λ) was used to train network weights at every time step t. The weight updates have the general form of $w_t = w_t + \alpha g_t$ where g_t is a λ-discounted running update over $0, \ldots, t$. The hidden-output layer (oh) and the input-hidden layer (hi) updates are computed, respectively, as:

$$(g_t)_{oh} = \lambda \, (g_{t-1})_{oh} + \delta_o y_h \qquad (g_t)_{hi} = \lambda \, (g_{t-1})_{hi} + \delta_h z_i$$

where:

$$\delta_o = f'(v_o) \; e_o \qquad \delta_h = f'(v_h) \sum_o e_o w_{oh}$$

The quantities $f'(v_o)$ and $f'(v_h)$ are the transfer function derivatives evaluated at the induced local fields $v_o = \sum_h w_{oh} y_h$ and $v_h = \sum_i w_{hi} z_i$, respectively, where $y_h = \tanh(v_h)$ and z_i is the value of input node i (all values are from time t). At the beginning of each episode, all values of g are set to zero.

The error e at time t is a 3-element vector corresponding to the 3 output nodes o:

$$e_o = \begin{cases} r_{t+1} + \gamma V(s_{t+1}, a_{t+1}) - V(s_t, a_t) & \text{if} \quad o = a_t \\ 0 & \text{if} \quad o \neq a_t \end{cases}$$

where $\gamma = 0.975$ is the next-state discount factor, r_{t+1} is the reward at time $t + 1$, and a_t and a_{t+1} are the actions taken at times t and $t + 1$.

ESANN 2014 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 23-25 April 2014, i6doc.com publ., ISBN 978-287419095-7.
Available from http://www.i6doc.com/fr/livre/?GCOI=28001100432440.

3.3 Training

Training consisted of having the agent attempt to learn the TBU domain over 40,000 episodes, where an episode consists of one attempt at backing the truck into the loading dock. For each episode, the starting state of the trailer was randomly initialized such that the vertical position y was sampled from $U[-10, 10]$, and the trailer angle θ_T was sampled from $U[-1.5, 1.5]$. The initial horizontal position x was set to 160, and the cab angle θ_C was set to 0.0. This amounts to the trailer having a random vertical position and a random orientation. During training, an ϵ-greedy action selection policy was used ($\epsilon = 0.95$), where exploitative actions are taken $100 \cdot \epsilon\%$ of the time, and explorative (random) actions are taken $100 \cdot (1 - \epsilon)\%$ of the time.

The goal of the problem was relaxed slightly such that the truck was considered to be at the loading dock in the correctly orientation if $\sqrt{x^2 + y^2} \leq 3$ and $\theta_T \leq 0.1$. When the trailer satisfied this condition, a reward of $r = +10$ was provided to the agent. When the truck was outside of this region, a reward was provided to the agent based on the trailer position and orientation, which took the form of: $r = -0.03 \cdot x^{0.6} - 0.002 \cdot |y|^{1.2} - 0.1 \cdot |\theta_T| + 0.4$. An episode was terminated if the cab jack-knifed ($\theta_C > \frac{\pi}{2}$), if the trailer angle became large ($\theta_T > 4\pi$), or if the front of the cab or the rear of the trailer exited the domain boundaries. In any of these cases, a penalty of $r = -0.1$ was provided to the agent, and the number of time steps in these episodes was set to the maximum number of time steps ($T_{max} = 300$). Episodes could also be terminated if the truck was successfully backed into the loading dock or if the T_{max} was reached.

Training performance was assessed using two types of metrics. The first metric was a moving average of the number of time steps in each episode using a 500-episode moving window. The second metric were moving proportions of the episode termination types, also using a 500-episode moving window. Empirical training convergence was assessed using three criteria that looked at the moving proportion of when the goal was reached: 1) the level must have been greater than 0.95; 2) the range must have been less than 0.01; and 3) the absolute value of the slope must have been less than 1×10^{-6}.

4 Results

The empirical convergence criteria were satisfied in 27,600 episode (Figure 2). The level, range, and slope of the moving proportion of reaching the goal was 0.992, 0.01, and -8.66×10^{-6}, respectively. At convergence, the 500-episode moving average of the number of time steps to the loading dock was 71.26. Figure 3 shows example trajectories of the truck backing into the loading dock from various starting locations and orientations. During these evaluations, a pure exploitative action selection policy was used ($\epsilon = 1.0$).

ESANN 2014 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 23-25 April 2014, i6doc.com publ., ISBN 978-287419095-7.
Available from http://www.i6doc.com/fr/livre/?GCOI=28001100432440.

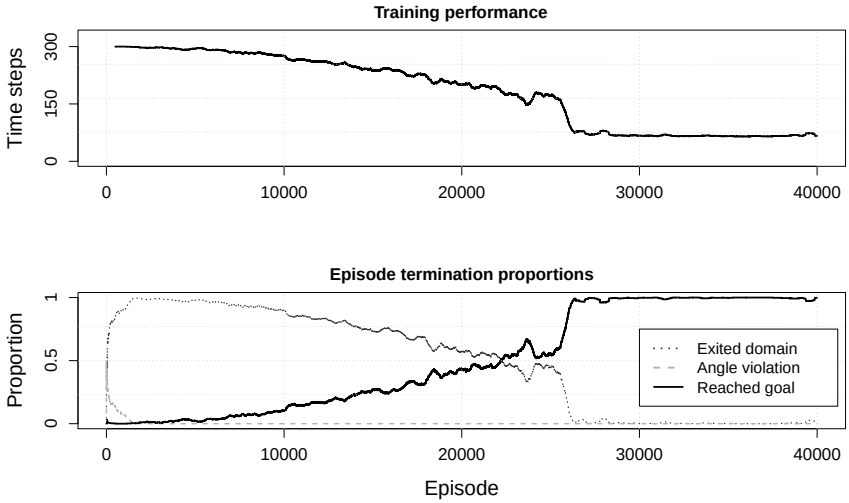

Fig. 2: Performance during training. The top plot shows the 500-episode moving average of the number of time steps per episode. The bottom plot shows the moving proportions of three termination types (termination types not shown were negligible).

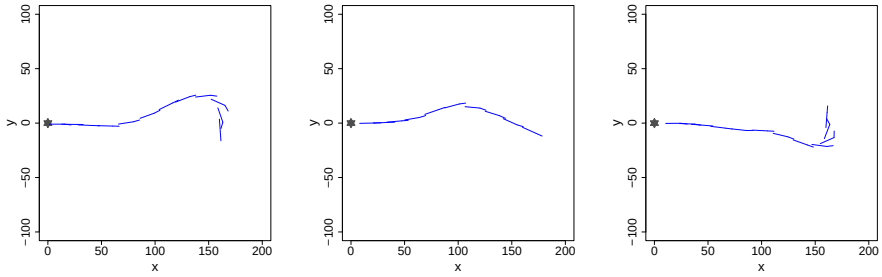

Fig. 3: Example test trajectories after training from random starting locations and orientations. The truck is depicted by the hinged lines, and the goal is indicated by the star located at (0,0).

5 Discussion

This work successfully applied one of the basic reinforcement learning algorithms to a real world control problem using a small neural network and common training procedures. This problem had some simplifications for the initial position/orientation constraints, though this problem is still general enough to be considered realistic such that the initial vertical position of the truck and its orientation were sampled over relatively wide ranges.

To further increase the accuracy and generalizability of this problem, a se-

ESANN 2014 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 23-25 April 2014, i6doc.com publ., ISBN 978-287419095-7.
Available from http://www.i6doc.com/fr/livre/?GCOI=28001100432440.

quential training procedure may be required. Such a procedure would begin with a training scheme as described here to seed future training runs that may have a stricter goal threshold or looser initial conditions. Training converged at 27,600 episodes, and this is admittedly an unrealistic number of trials for a real world implementation to perform. However, the implementation in this work could be used as an *in silico* training scheme, which could then be ported to a real truck to refine its real world performance. Additionally, other reinforcement learning algorithms, such as Q-learning [10] may learn this task more efficiently.

The parameters and settings used in this implementation were based on those generally used with TD(λ) and neural networks, and some trial and error was required to find appropriate settings. These parameters largely include the structure of the neural network, learning rates α, ϵ, λ, and γ, and it is likely that there are more optimal settings for this task. The parameters used herein are relatively robust, however, slight modifications of these parameters would occasionally result in learning runs that did not empirically converge. Exploring parameter subregions in which learning converges more rigorously is the focus of ongoing work.

References

[1] K. Doya, Reinforcement learning in continuous time and space, *Neural Computation*, vol. 12, pp. 219–245.

[2] C. J. Gatti, J. D. Linton, and M. J. Embrechts. A brief tutorial on reinforcement learning: The game of Chung Toi. In *Proceedings of the 19th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, 2011.

[3] C. J. Gatti and M. J. Embrechts. Reinforcement Learning with Neural Networks: Tricks of the trade. In P. Georgieva, L. Mihayolva, and L. Jain (eds.), *Advances in Intelligent Signal Processing and Data Mining*, pp. 275–310, Springer-Verlag, 2012.

[4] C. J. Gatti, M. J. Embrechts, and J. D. Linton. An empirical analysis of reinforcement learning using design of experiments. In *Proceedings of the 21st European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, 2013.

[5] A. W. Moore, Efficient memory-based learning for robot control. PhD Thesis, University of Cambridge, 1990.

[6] A. Y. Ng, H. J. Kim, M. I. Jordan, and S. Sastry. Inverted autonomous helicopter flight via reinforcement learning. In *International Symposium on Experimental Robotics*, MIT Press, 2004.

[7] D. Nguyen and B. Widrow, Neural networks for self-learning control systems, *IEEE Control Systems Magazine*, pp. 18–23, 1990.

[8] D. Nguyen and B. Widrow, The truck backer-upper: An example of self-learning in neural networks. In W. T. Miller, R. S. and P. J. Werbos, editors, Neural Networks for Control, MIT Press, 1990.

[9] M. Schoenauer and E. Ronald, Neuro-genetic truck backer-upper controller. In *Proceedings of the 1st IEEE Conference on Evolutionary Computation*, vol. 2, pp. 720–723, 1994.

[10] R. S. Sutton and A. G. Barto. *Reinforcement Learning*. MIT Press, 1998.

[11] H. Vollbrecht, Hierarchical reinforcement learning in continuous state spaces. PhD Thesis, University of Ulm, 2003.