ESANN 2013 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2013, i6doc.com publ., ISBN 978-2-87419-081-0.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100131010.

An empirical analysis of reinforcement learning using design of experiments

Christopher J. Gatti[1], Mark J. Embrechts[1], and Jonathan D. Linton[2] *

[1] Rensselaer Polytechnic Institute – Dept. of Industrial and Systems Engineering
Troy, NY - USA

[2] University of Ottawa - Telfer School of Management
Ottawa - Canada

Abstract. This study uses a design of experiments approach to understand the behavior of a neural network to learn the mountain car domain using the TD(λ) algorithm. A large experiment is first performed to characterize the probability of empirical convergence based on three parameters of the TD(λ) algorithm (λ, γ, ϵ), and a logistic regression model is fitted to this data. A detailed analysis of the parameter subspace finds that, upon convergence, these parameters significant affect convergence speed and mean performance, though performance differences are minimal.

1 Introduction

Reinforcement learning has had a handful of successes in challenging domains, including Backgammon [8] and helicopter control [6]. However, this learning method has not had nearly the success of other machine learning approaches, and this may be due to our limited understanding of the complex interactions between the learning algorithms, functional representations, and domain characteristics. Theoretical analysis of reinforcement learning is limited to rather simplistic scenarios [9], and this motivates the use of empirical methods to understand the behavior of reinforcement learning [1, 2, 7]. However, empirical studies often use simple parameter studies and assess effects by comparative observations [7], which cannot easily reveal parameter interactions. A more efficient and statistically rigorous approach is to use formal design of experiments approaches [4]. The purpose of this study is to use a design of experiments approach to understand the effects of three parameters of the TD(λ) algorithm when using a neural network to learn the mountain car domain.

2 Methodology

This study is based on coupling a reinforcement learning application, the mountain car domain, with an experimental design, and this section describes each component of this experimental system. We use this work as a proof-of-concept in applying design of experiments to understand the performance of reinforcement learning. Consequently, we restrict this work to a relatively simple domain

*The authors acknowledge the support of the National Sciences and Engineering Research Council of Canada.

ESANN 2013 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2013, i6doc.com publ., ISBN 978-2-87419-081-0.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100131010.

and a fundamental model-free learning algorithm, however these methods can be used to understand other learning algorithms in additional domains.

2.1 Mountain car domain

The mountain car domain [5] which places a car in a valley, where the goal is to get the car to drive out of the valley. The car's engine is not powerful enough for it to drive out of the valley, and the car must instead build up momentum by successively driving up opposing sides of the valley. The state of the car is defined by its position $x \in [-1.2, 0.5]$ and its velocity $\dot{x}\ in[-1.5, 1.5]$, and the goal is located at $x = 0.5$. At the beginning of each episode, the x is uniformly randomly sampled from $[-1.2, 0.5]$ and $\dot{x} = 0$. The dynamics of the car follow:

$$x_{t+1} = x_t + \Delta t \dot{x}_t \qquad \dot{x}_{t+1} = \dot{x}_t + \Delta t \left(-9.8\ m\ \cos(3x_t) + \frac{f}{m}\ a - \mu\ \dot{x}_t \right)$$

where $\Delta t = 0.01$ is the time step, $m = 0.02$ is the car's mass, $f = 0.2$ is the engine force, and $\mu = 0.5$ is a friction coefficient. The variable a represents the action taken by the agent, where $a = -1$ for driving backwards, $a = 0$ for neutral, and $a = 1$ for driving forwards. At every time step that the car has not reached the goal, the agent receives a reward (i.e., penalty) r equal to x. When the car reaches the goal, the agent receives a reward of 1, and the episode ends.

2.2 Agent representation

A three-layer neural network is used to represent the agent and to learn the value function $V(s_t, a_t)$, which represents the value of pursuing action a_t while in state s_t. The network has 2 inputs, corresponding to the state $s_t = [x_t, \dot{x}_t]^T$, 21 hidden nodes, and 3 output nodes that represent the values of the three actions. The hidden and output layers use tanh and linear transfer functions, respectively. Each time a network is created, new weights are initialized by uniform random sampling over $[-0.1, 0.1]$. The learning rates α are initialized by layer using a heuristic similar to that described in [3]. 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 $\frac{1/(4 \cdot 500)}{min(\alpha)}$, which is based on the maximum number of time steps (500), resulting in $\alpha_{oh} = 0.0028$ and $\alpha_{hi} = 0.0005$.

For each experimental run, a network is trained for a maximum of 2000 episodes, where each episode consists of one attempt at trying to get the car to the goal. Network weights are updated at every time step using the temporal difference algorithm (TD(λ)) [7]. The general form of the weight updates at time t follows $w_t = w_t + \alpha g_t$ where g_t is a λ-discounted update over all time steps up to and including t. The hidden-output layer (oh) and the input-hidden layer (hi) updates are computed, respectively, as:

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

ESANN 2013 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2013, i6doc.com publ., ISBN 978-2-87419-081-0.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100131010.

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 for 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 γ 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 $t+1$.

For each experimental run, a newly initialized network is trained and tested in the mountain car domain. A network is considered to have converged if it satisfies both training and testing convergence criteria. Training convergence requires two conditions: 1) the range of the 200-episode moving average of the number of time steps for the car to reach the goal is less than 10 time steps, and 2) all episodes in the last 200 episodes require less than the maximum number of allowed time steps (500), otherwise a maximum of 2000 episodes is allowed. During training, the agent pursues an ϵ-greedy action selection procedure ($\epsilon = [0, 1]$) in which it selects the action with the greatest predicted state value $100\epsilon\%$ of the time, and it selects a random action $100(1-\epsilon)\%$ of the time. Following training, agent performance is tested using 200 test episodes in which it uses a pure exploitative policy (i.e., $\epsilon = 1$). Testing convergence requires that the car reach the goal in all 200 test episodes, and performance was quantified by the average number of time steps required to reach the goal over the 200 test episodes.

2.3 Experimental design and analysis

The goal of this work is to *understand the effects* of λ, γ, and ϵ with respect to learning convergence and performance; this work is not aimed at optimizing (i.e., tuning) parameter settings. This study was based on a single experimental design with a two-stage analysis. The first analysis is aimed at assessing network convergence over a large parameter space. A full factorial experiment ($\mathcal{D}_1$) is run with the following continuous level settings for each parameter: λ over $[0.1, 0.9]$ incremented by 0.1, γ over $[0.95, 0.99]$ incremented by 0.01, and $\epsilon = \{0.7, 0.8, 0.9\}$. This experiment therefore consists of 135 factor-level combinations, with 10 replications at each factor-level combination. The outcome for this experiment is a binary variable indicating (empirical) convergence; recall that convergence requires that the network converge during both training and testing. A logistic regression (LR) model is then created to estimate the probability of convergence based on λ, γ, and ϵ in $\mathcal{D}_1$.

The second analysis aims to determine the effects of λ, γ, and ϵ on performance over a smaller parameter space in which the network frequently converges. The smaller parameter space $\mathcal{D}_2$ is a subset of and is extracted from $\mathcal{D}_1$

ESANN 2013 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2013, i6doc.com publ., ISBN 978-2-87419-081-0.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100131010.

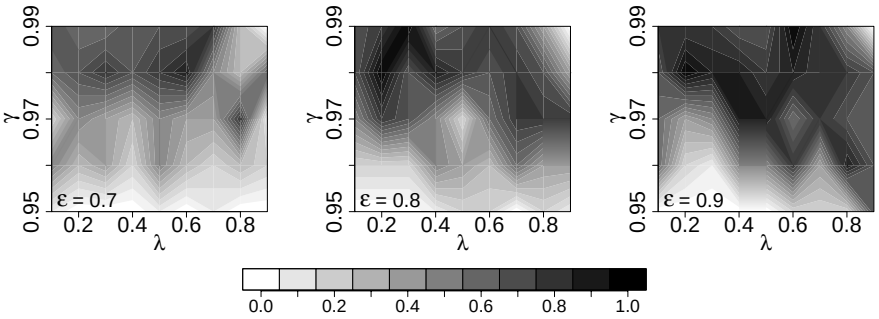

Fig. 1: Empirical probability of convergence over $\mathcal{D}_1$ for values of ϵ.

($\mathcal{D}_2 \subset \mathcal{D}_1$), where $\mathcal{D}_2$ consists of the following level settings: $\lambda = \{0.6, 0.7, 0.8\}$, $\gamma = \{0.7, 0.8, 0.9\}$, and $\epsilon = \{0.97, 0.98, 0.99\}$. These factor levels were chosen after assessing network convergence over $\mathcal{D}_1$. This design is a 3×3 full factorial design, with 10 replications at each of the 27 factor-level combinations. Analysis of variance (ANOVA) with Type II sums of squares is used to determine if λ, γ, and ϵ (and their interactions) has significant effects on the convergence speed (i.e., episode at which training converged) and on the mean testing performance. Non-convergent runs are qualified as *undefined* responses, as opposed to *missing* data, and these runs are removed from the data for the analysis, resulting in unbalanced groups and the need for Type II sums of squares.

3 Results

Experimental design $\mathcal{D}_1$ resulted in 77.85% (1051/1350) of the runs converging during training, and 48.59% (656/1350) converging based on both training and testing convergence criteria. The proportion of times that unique factor-level combinations converged ranged from 0/10 to 10/10, confirming that some regions of the parameter space that are clearly better than others. Figure 1 shows the empirical probabilities of convergence over $\mathcal{D}_1$. A LR model was created to estimate network convergence using linear, quadratic, and interaction terms (Table 1), and nearly all terms have statistically significant coefficients. The LR model was used because it provides a compact functional form for predicting convergence in this application, though other function approximators, such as neural networks, could be used to model the convergence probability.

Experimental design $\mathcal{D}_2$ resulted in 98.89% (267/270) of the runs converging during training, and 67.78% (183/270) converging based on both training and testing convergence criteria. Figure 2 shows boxplots for the convergence episode and mean testing performance versus the levels of λ, γ, and ϵ. Table 2 shows the ANOVA models for the logarithm of these responses. Parameters λ and γ were found to have a significant interaction effect on convergence speed, and although the interpretation of their main effects is therefore not straightforward, it is likely they have significant main effects based on their low p-values; ϵ also has a significant main effect. For the mean performance after convergence, γ is

ESANN 2013 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2013, i6doc.com publ., ISBN 978-2-87419-081-0.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100131010.

Model term	Estimate	Standard error	Estimate p-value	Deviance reduction	Deviance p-value
Intercept	-3497.89	432.65	**<0.001**	—	
λ	-544.71	202.52	**0.007**	0.545	0.460
γ	7381.65	863.16	**<0.001**	228.685	**<0.001**
ϵ	-391.58	153.81	**0.011**	33.398	**<0.001**
λ^2	-4.26	1.19	**<0.001**	11.763	**<0.001**
γ^2	-3896.11	439.18	**<0.001**	79.475	**<0.001**
ϵ^2	-5.76	13.646	0.67	0.061	0.805
$\lambda : \gamma$	557.82	208.00	**0.007**	59.748	**<0.001**
$\lambda : \epsilon$	860.75	251.49	**<0.001**	5.655	**0.017**
$\gamma : \epsilon$	412.61	156.08	**0.008**	1.299	0.254
$\lambda : \gamma : \epsilon$	-876.96	258.31	**<0.001**	11.720	**<0.001**
Residual deviance:	1438.1	($df = 1339$)			
Hosmer-Lemeshow GOF:	$p = 0.318$				

Table 1: Summary of the logistic regression model.

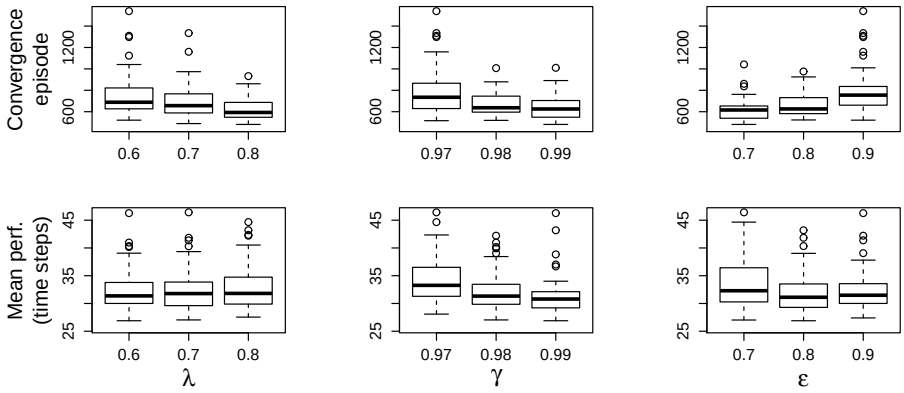

Fig. 2: Boxplots of episode of performance measures versus λ, γ, and ϵ.

the only parameter that had significant effects, and $\gamma = 0.99$ resulted in the best (smallest) mean performance (Fig. 2).

4 Discussion and Conclusions

This study employs design of experiments and statistical analysis to aid in understanding the behavior of TD(λ) in a specific domain. Experimental design $\mathcal{D}_1$ shows that the network empirically converges to a stable solution in a fairly small region of the parameter space, however, convergence is not guaranteed. Experimental design $\mathcal{D}_2$ indicates that significant effects are of low order. While all parameters have an effect on convergence speed, only γ has a significant effect on mean performance (though the effect is only ~ 4 time steps), and λ and ϵ do not affect the quality of the solution. In other words, while λ, γ, and ϵ and their interactions significantly affect convergence, they have little or no practical impact on mean performance. A natural extension of this work is to use addi-

ESANN 2013 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2013, i6doc.com publ., ISBN 978-2-87419-081-0.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100131010.

Convergence episode

Model term	SS	F-ratio	p-value
λ	0.207	48.864	**<0.001**
γ	0.231	54.603	**<0.001**
ϵ	0.295	69.557	**<0.001**
$\lambda : \gamma$	0.037	8.812	**0.003**
$\lambda : \epsilon$	0.016	3.752	0.054
$\gamma : \epsilon$	0.005	1.164	0.282
Residuals	0.747		

Mean performance

Model term	SS	F-ratio	p-value
λ	0.000	0.000	0.996
γ	0.043	20.275	**<0.001**
ϵ	0.005	2.368	0.126
Residuals	0.381		

Table 2: Analysis of variance for the logarithm of the two response variables; the colon between variables indicates a variable interaction (SS = sums of squares).

tional design of experiments methods, such as response surface methodologies, to optimize parameter settings.

From a design of experiments perspective, this experiment has a unique characteristic such that some runs may not converge, and this scenario has received little or no attention in the literature. These unique outcomes motivated the use of the sequential analysis in order to separate convergence and parameters effects. Experimental design $\mathcal{D}_2$ focused on a small parameter space that converged very frequently, though it did not always converge. Furthermore, caution should be used when extrapolating the results to parameters outside of the ranges used in this study, as severe nonlinearities were observed at the parameter space edges (e.g., $\lambda = 0.01$ or 0.99, or $\gamma = 1$). This study investigated the effects of the primary variables of the TD(λ) algorithm, though the learning rate α likely also has an effect on learning, and this could be included in future work. Finally, the extensibility of the findings presented herein to different representations, learning algorithms, or domains is unknown, and more exhaustive studies are needed to form generalizable conclusions.

References

[1] S. Bhatnagar, R. Sutton, M. Ghavamzadeh, and M. Lee. Natural actor critic algorithms. *Automatica*, 45:2471–2482, 2009.

[2] C. J. Gatti, M. J. Embrechts, and J. D. Linton. Parameter settings of reinforcement learning for the game of Chung Toi. In *IEEE International Conference on Systems, Man, and Cybernetics*, pp. 3530–3535, Anchorage (Alaska), 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] D. C. Montgomery, *Design and Analysis of Experiments*. Wiley, New York, 2008.

[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 *Intl. Symp. on Experimental Robotics*, MIT Press, 2004.

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

[8] G. Tesauro. Temporal difference learning and TD-Gammon, *Communications of the ACM*, 38:58–68, 1995.

[9] J. N. Tsitsiklis and B. Van Roy. Feature-based methods for large scale dynamic programming. *Machine Learning*, 22:59–94, 1996.