

Solving Survival Gridworld Problem Using Hybrid Policy Modified Q-Based Reinforcement

Vince Jebryl Montero*, Woo-Young Jung*, Yong-Jin Jeong *

Abstract

This paper explores a model-free value-based approach for solving survival gridworld problem. Survival gridworld problem opens up a challenge involving taking risks to gain better rewards. Classic value-based approach in model-free reinforcement learning assumes minimal risk decisions. The proposed method involves a hybrid on-policy and off-policy updates to experience roll-outs using a modified Q-based update equation that introduces a parametric linear rectifier and motivational discount. The significance of this approach is it allows model-free training of agents that take into account risk factors and motivated exploration to gain better path decisions. Experimentations suggest that the proposed method achieved better exploration and path selection resulting to higher episode scores than classic off-policy and on-policy Q-based updates.

Key words : Reinforcement learning, Deep Q-learning, Value-based reinforcement, Gridworld, Hybrid Policy

1. Introduction

Reinforcement learning (RL) is an area of machine learning concerned with sequential decision making process that aims to maximize potential gains. It addresses the problem of an agent learning to act in an environment in order to maximize cumulative rewards [2]. Deep reinforcement learning (DRL) is the result by combining deep learning and RL [1]. DRL opens up a solution to a wide range of complex decision making tasks that were previously unattainable for a machine [1]. In model-free reinforcement learning, the agent actively collects experiences by interacting with the environment following a given policy. These experiences are then evaluated based on

the stimulus received, either a reward or a penalty to encourage or discourage sequence of actions in a particular situation. The state given the evaluated stimulus forms the training data for the estimator (neural network) to learn successful control policies.

With the environment called gridworld, provides a natural environment for the agent. The gridworld consists of a two-dimensional grid of cells in which the agent and objects occupy one cell of the grid [4]. The agent is often given a task or challenge in the gridworld where it can move to the four adjacent directions or interact with objects in a cell. Gridworld tasks can offer difficulties that poses a challenge in RL despite its simplistic nature.

* Dept. of Electronics and Communications Engineering, Kwangwoon University

★ Corresponding author

E-mail : yjjeong@kw.ac.kr, Tel : +82-2-940-5551

※ Acknowledgment

This work was supported by Kwangwoon University and by the MISP Korea under the National Program for Excellence in SW (2017-0-00096) supervised by IITP.

Manuscript received Oct. 30, 2019; revised Dec. 2, 2019; accepted Dec. 11, 2019.

This is an Open-Access article distributed under the terms of the Creative Commons Attribution Non-Commercial License (<http://creativecommons.org/licenses/by-nc/3.0>) which permits unrestricted non-commercial use, distribution, and reproduction in any medium, provided the original work is properly cited.

This paper explores a model-free value-based approach for solving a particular gridworld problem: the survival gridworld problem that may involve taking risks in order to complete the task and gain better rewards. The proposed method introduces a parametric linear rectifier and a motivational discount to the classic Q-based update equation. It also involves utilization of hybrid on-policy and off-policy control to gain better control policies.

II. Background

The general RL problem is presented as a discrete time step in stochastic control where the agent interacts with the environment following a Markov decision process (MDP) [1], [4], [5]. The MDP is formalize by the tuple $\langle S, A, T, R \rangle$ where it involves the state space S , the action space A , the transition function $T: S \times A \rightarrow \Delta S$, and the reward function $R: S \times A \rightarrow \mathbb{R}$ [1]–[5]. The agent starts at an initial state $s_0 \in S$. Then, at each time step it observes the current input state $s_t \in S$ and takes an action $a_t \in A$ drawn from a distribution in the transition function $T(s_t, a_t)$ and receives a reward $R(s_t, a_t)$ [1]–[5]. The step process continues until the agent reaches a terminal state. Then, after which the process restarts again.

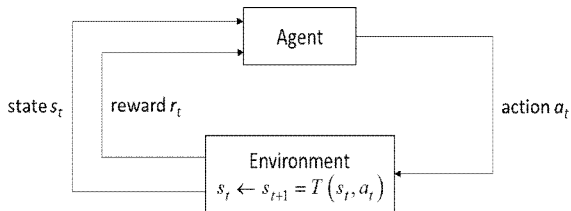


Fig. 1. Agent to environment interaction in MDP.

The goal of the agent is to maximize the reward from each state s_t . The quality of action value given by $Q^*(s, a) = E[R_t | s_t = s, a_t = a, \pi]$ which is the expected return from selecting an action a in state s following the action policy π where the sum of discounted rewards r_t per time step is described by $R_t = \sum_{t'=t}^T \gamma^{t'-t} r_{t'}$; $\gamma \in [0, 1]$ [1], [5], [6].

Optimizing the quality of action value function with respect to the Bellman equation [5] assuming the optimal value of $Q^*(s_{t+1}, a_{t+1})$ for the state s_{t+1} in the next time step is known for all actions a_{t+1} will yield the expected value for the current time step described by $Q^*(s_t, a_t) = r_t + \gamma \max_a Q^*(s_{t+1}, a)$ [3].

III. Deep Reinforcement Learning

Value-based approach in deep reinforcement learning aims to build a value function that defines a control policy [1]. The estimator (neural network) is used as a value function to predict the next action given the input state. Training a deep learning agent comes in three phases. First, is collecting experiences through agent to environment interactions defined by a control policy. Second, preparing training samples by processing the state and Q-value pairs. Raw Q-value output from the estimator is used to create the training labels Y^Q for the input state by utilizing the update formula derived from the MDP:

$$Y_k^Q = r_t + \gamma \max_a Q(s_{t+1}, a; \theta_k) \quad (1)$$

where θ_k refers to parameters (network weights) that defines the Q-values at the k^{th} iteration while the time step t only refers to the episodic roll-outs in the collected experiences. Third, optimizing the estimator using the state- Y^Q pair as training data and with the loss function defined by:

$$L = (Q(s, a; \theta_k) - Y_k^Q)^2 \quad (2)$$

The control policy for interacting with the environment is defined by the epsilon-greedy policy (ϵ -greedy). This is derived from the exploration-exploitation dilemma where at first the experiences is collected by random walk (taking random action). Then, the chance of random action is decreased gradually as the estimator learns to

make decisions in the environment.

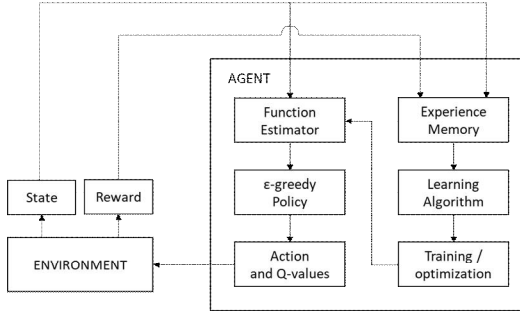


Fig. 2. Diagram of deep RL method.

IV. Environment

Survival gridworld is a two-dimensional grid environment where obstacles and rewards are distributed between the start-point and the end-point. The task of the agent is to navigate a path from the start-point to end-point while keeping a positive lifeline or energy meter e . The agent is allowed to move in one of the four adjacent directions. Each movement in the gridworld costs a fixed energy defined by the transition cost δs . Also, obstacles (negative scalar) decrease the energy while rewards (positive scalar) increase it. The condition for failure is when $e \leq 0$ and the successful terminal state is when the agent reaches the end-point cell. At the end of the episode the score is computed:

$$score = \begin{cases} \max(1, e - e_0 + 1), & \text{if successful} \\ 0, & \text{else} \end{cases} \quad (3)$$

where e_0 is the initial energy.

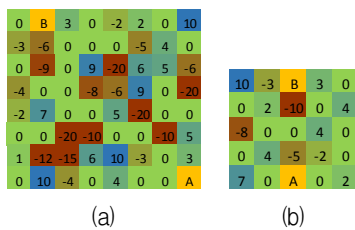


Fig. 3. Survival gridworld sample. Cells A and B are the start-point and end-point.

(a) 8x8 gridworld. (b) 5x5 gridworld.

V. Proposed Method

1. Estimator Design

A convolution neural network (CNN) is used as value function estimator. The CNN design used is shown in Table 1. The design for the convolutional layers is made to effectively extract significant features from the state input while the fully connected layers are optimized for the number of Q-value outputs for each action. The dropout units in the fully connected layers help to stabilize perturbations during training caused by varying estimates of the Q-values. The state input to the estimator is comprised of four channels described as follows: Current state – 5×5 cell cut-out of the surrounding relative to the agent's position. Previous state – 5×5 cell cut-out of the previous surrounding relative to the agent's previous position. Location state – shows the start-point and end-point with the agent's current position as well as the agent's four previous positions. Energy state – shows the representation of the agent's lifeline as a two-dimensional energy bar. The current and previous gridworld state ensures that the estimator can perceive the changes and movement. The location state directs the trajectory while the energy state helps to hint at the current lifeline.

Table 1. Estimator design

Input Layer	Input : 16x16 x4 channels
Convolution Layer	Conv. 1 : filter 4x4x16, ReLu, MaxPool 2x2 Conv. 2 : filter 4x4x32, ReLu, MaxPool 2x2
Fully Connected	Dense 1 : 64 units, ReLu, Drop rate=0.20 Dense 2 : 128 units, ReLu, Drop rate=0.40
Output Layer	Dense output : 4 units only (corresponds to four actions)

2. Reward Function Design

The design of the reward function is very crucial for the agent's successful training. The reward function is as follows:

$$r_t = \begin{cases} +1, & \text{if successful} \\ -1, & \text{if failed} \\ -0.3, & \text{if boundary bump} \\ r_t^*, & \text{if non-terminal} \end{cases} \quad (4)$$

$$r_t^* = \begin{cases} \frac{g^{n,m}}{n^+} L^+ \sigma(e_t), & \text{if } g^{n,m} > 0 \\ \frac{g^{n,m}}{n^-} (1 - L^- \sigma(e_t)), & \text{else} \end{cases} \quad (5)$$

where $g^{n,m}$ is the value of the grid cell, n^+ is the maximum cell value plus a small positive integer, n^- is the minimum cell value plus a small negative integer, $L^+, L^-: 0 < L^+, L^- \leq 1$; $L^+ > L^-$ is the reduction factor. The energy state multiplier is described as:

$$\sigma(e_t) = \frac{1}{1 + \exp\left(\frac{-k}{v}(e_t - 0.25v)\right)} \quad (6)$$

where e_t is the current lifeline or energy state, k is the scale for the minimum value of $\sigma(e_t)$ such that $\min(\sigma(e_t)) \approx \frac{100}{k}$, and v defines the value for asymptote approach.

The reward function may look a bit complicated but the intuition behind it is simple. The higher the current energy state e_t the higher the gain and the lower the risk. In the survival gridworld problem, often risks are necessary to attain higher rewards.

3. Learning Algorithm

The learning algorithm utilizes a hybrid off-policy and on-policy control update. Off-policy updates are Q-value updates that is not based on the agent's experience from a series of actions such as in eq. 1. In eq. 1 instead on using the value $Q(s_{t+1}, a_{t+1}; \theta_k)$ for the future action reward, it uses the maximum value $\max_a Q(s_{t+1}, a; \theta_k)$. Another example is supervised learning from human expert moves in which the updates are not based on the agent's control policy. Contrary to off-policy, on-policy updates are based on the agent's series of actions.

The utilization of a hybrid policy update comes

from the intuition that off-policy has a greedy behavior such that it only focuses on attaining the goal at any cost. It means that off-policy agents are risk takers and it is more concerned in the final outcome. While, on-policy agents are safe players which can be good in exploring the rewards in between, but less likely to converge on a better path to the final goal given more risks. Combining the two policies result in better convergence to the final goal while gaining more rewards in between.

The off-policy and on-policy update equations are described respectively as follows:

$$Y_k^{Qv} = r_t + \omega \max_a Q(s_{t+1}, a; \theta_k) \quad (7)$$

$$Y_k^{Q\pi} = r_t + \omega \text{ReLu}_\rho [Q(s_{t+1}, a_{t+1}; \theta_k)] \quad (8)$$

where the parametric linear rectifier is defined as $\text{ReLu}_\rho(x) = \max(x, \rho x)$. The motivational discount is:

$$\omega = \gamma [(1-\eta)M + \eta]; 0 < \gamma, \eta \leq 1 \quad (9)$$

where M is the motivation function.

On the presented case, M is the ratio of the Euclidian distance between the current position and the end-point over the distance from start-point to end-point. The intuition behind the motivational discount is that, the nearer the traversed path to the end-point, the better the future rewards. This has also the effect of slightly raising the update values at a later training period which balances out the decrease in learning rate of the estimator. The parametric linear rectifier drastically shrinks the effects of any major obstacles following a good reward.

Table 2. Hybrid update rule.

$Y_k^{Q\pi}$	<ol style="list-style-type: none"> 1. Use at the start of the training period 2. After every testing period: When success rate is higher than a threshold use on-policy update
Y_k^{Qv}	<ol style="list-style-type: none"> 1. After every testing period: When success rate is lower than a threshold use off-policy update

This encourages the agent to venture high risk areas. The rules on when to apply on-policy Y_k^{Qr} and off-policy Y_k^{Qv} update are described in Table 2.

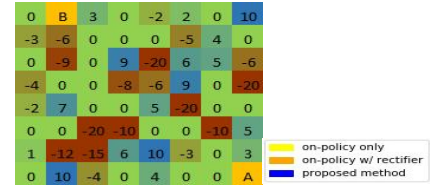
VI. Experimentation

To analyze and evaluate the performance of the proposed method compared to classic Q-based algorithms (off-policy; deep Q-learning and on-policy: deep Sarsa algorithm [5]), carefully tuned survival gridworlds are presented. These gridworlds are tuned to have several paths leading to the end-point. The paths include low-risk low-rewards, high-risk high-rewards, and cross-over paths (cross-over from low-risk to high-risk) with several off-path rewards (higher chance to be ignored).

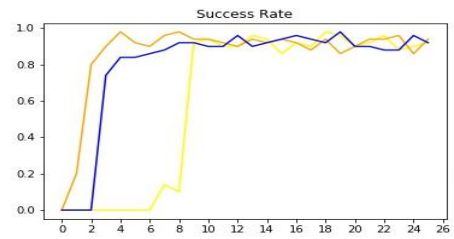
In experiment 1, the gridworld in Fig. 4a is designed to test the effects of on-policy rectification (Eq. 8) where two main paths are constructed, low-risk low-rewards and high-risk high-rewards, and one cross-over path. In experiment 2, the gridworld in Fig 4e is designed for exploration testing where several opening paths and cross-over are constructed. Also, several off-path rewards are placed within cross-over paths to confuse the agent. In experiment 3, the gridworld in Fig. 4i is designed for path selection testing where several paths are constructed in a way that once an agent got used to taking a certain low rewards path, it will be harder to shift to another path (might be a high rewards path). Also, off-path rewards are placed to test for off-path exploration.

The following metrics, success rate and total rewards, are used to analyze the behavior of the agent under a particular learning algorithm. Successful episode is when the agent traverses the start-point to end-point and the total rewards is the sum of all non-zero cells visited by the agent. The metric, episode score, is used to summarize the overall performance of the

agent, combining success rate, total rewards, and assessment of the travelled path. Eq. 3 is used to compute the episode score.



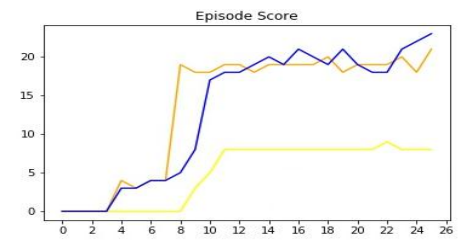
(a)



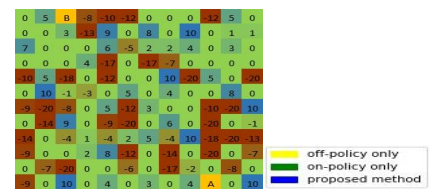
(b)



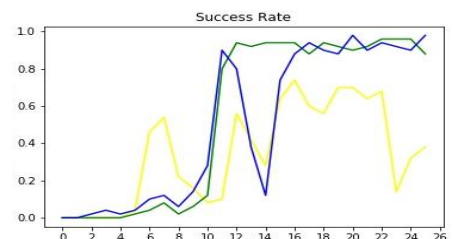
(c)



(d)



(e)



(f)

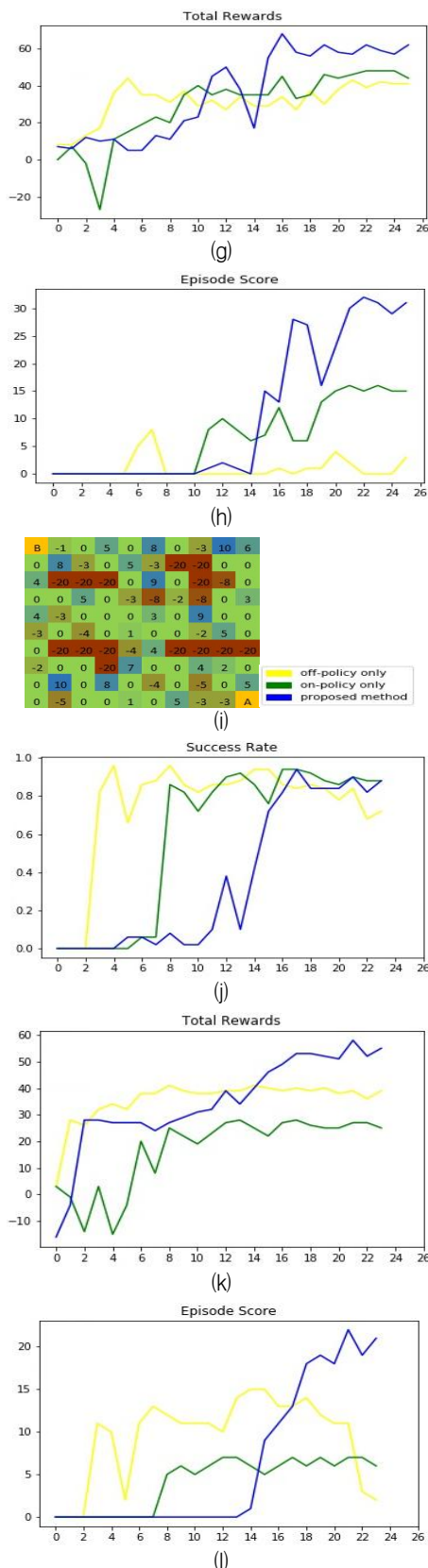


Fig. 4. Test results for every training iteration. (a, b, c, d) Experiment 1 results. (e, f, g, h) Experiment 2 results. (i, j, k, l) Experiment 3 results. The x-axis of the plots correspond to a factor of 1×10^3 episodes.

During experimentation, the agent is trained on different learning methods. After every training iteration, the agent is evaluated by playing 50 episodes where the results are presented in Fig 4. The results in experiment 1 as illustrated in Fig. 4b–4d shows the effect of rectification. Rectified on-policy method performs much better than classic on-policy method based on the episode score as presented in Fig. 4d. Based on the positive reward distribution shown in Fig. 4a, it can be deduced that the rectifier reduces the declining effects of venturing into high risk areas. Also, to compare, the proposed method achieved higher score which means better exploration than using on-policy only.

In experiments 2 and 3, the effects of motivational discount ω is more pronounced in more complex environments shown in Fig. 4e and 4i. The delayed stable success period of the proposed method shown in Fig. 4f and 4j is due to reduced step rewards in early exploration. But, this allows the agent to strengthen actions that provide better gains at later exploration. Thus, in Fig. 4g and Fig. 4k, it can be seen that classic off-policy and on-policy methods has no further progress as opposed to the proposed method which at later episodes gained further positive rewards. Moreover, in experiment 2, the proposed method achieved better path exploration which resulted in scoring 1.70x better than the combined scores of the two classic methods as shown in Fig 4h. Also, in experiment 3, the proposed method, achieved better path selectivity and off-path exploration which resulted in scoring more than 1.40x better than each of the two classic methods as shown in Fig 4l.

The role of the hybrid update rule is to stabilize the effect of ω as fluctuations in future rewards produce variances that affect the learning of the estimator. Such effect can be seen in Fig. 4h and Fig. 4j where there are sharp dips in the success rate of the proposed method. The designed experiments demonstrated the general behavior of

the agent under the proposed method. From the experiment results, it can be deduced that similar behavior can be observed for other environments as long as M in Eq. 9 for ω is well defined relative to the target output. Overall, the proposed method achieved better scores at the end of the training period shown in Fig. 4d, Fig. 4h and Fig. 4l.

VII. Conclusion

This paper proposes a hybrid on-policy and off-policy updates to experience roll-outs using a modified Q-based update equation that introduces a parametric linear rectifier and motivational discount. Experimentations in section VI suggest that the proposed method achieved better exploration and path selection resulting to higher episode scores than classic off-policy and on-policy Q-based updates. To which, the significance of this approach allows model-free training of agents that take into account risk factors and motivated exploration to gain better path decisions. This behavior is highly valued especially to the field of robotics where autonomous drones can be potentially trained for exploration tasks where risks are always present (such as unavoidable damages and early power exhaustion).

Future work includes applying the modified Q-based updates to policy-based methods such as actor-critic [1], [5] and asynchronous methods [6]. This will be significant in taking a step to solve the challenges presented in [4] which is a major milestone in AI reinforcement learning.

References

- [1] François-Lavet, Vincent et al. "An Introduction to Deep Reinforcement Learning," *Foundations and Trends in Machine Learning*, Vol.11, No.3-4, 2018. DOI: 10.1561/22000000071
- [2] Hessel, Matteo et al. "Rainbow: Combining Improvements in Deep Reinforcement Learning", *The 32nd AAAI Conference on Artificial Intelligence*,

pp.3215-3222, 2018.

- [3] Mnih, Volodymyr et al. "Playing Atari with Deep Reinforcement Learning," *NIPS Deep Learning Workshop 2013*. 2013.
- [4] Leike, Jan et al, "AI Safety Gridworlds," arXiv preprint arXiv : 1711.09883v2, (2017).
- [5] Sutton, R. S. and Barto, A. G. "Reinforcement Learning: An introduction 2nd Edition. Cambridge," Massachusetts : The MIT Press, 2018.
- [6] Mnih, Volodymyr et al. "Asynchronous Methods for Deep Reinforcement Learning," arXiv preprint arXiv: 1602.01783v2, 2016.

BIOGRAPHY

Vince Jebryl Montero (Member)



2014 : BS degree in Electronics Engineering, Ateneo de Davao University.
2017 ~ : Currently studying for combined MS and PhD course in Electronics and Communications Engineering, Kwangwoon University.

Woo-Young Jung (Member)



1991 : BS degree in Electronics and Communications Engineering, Kwangwoon University.
1996 : MS degree in Electronics and Communications Engineering, Kwangwoon University.

Current : PhD course in Electronics and Communications Engineering, Kwangwoon University.
2012 ~ : CTO in Innovative Research and Analysis Inc.
2013 ~ : CTO in EPSOLUTE Co. Ltd.

Yong-Jin Jeong (Member)



1983 : BS. degree in Control and Instrumentation Engineering. Seoul National University.
1995 : MS, and PhD degree in Electronics and Computer Engineering, University of Massachusetts, Amherst.

1983~1989 : Research Engineer, ETRI.
1995~1999 : Chief Researcher, Samsung Electronics.
1999~ : Professor, Dept. of Electronics and Communications Engineering, Kwangwoon University.