Zap Q-Learning and its applications in Arcade Learning Environment (ALE)

*Xuyin Zhu; Muchen Li; Xuanzhao Dong*

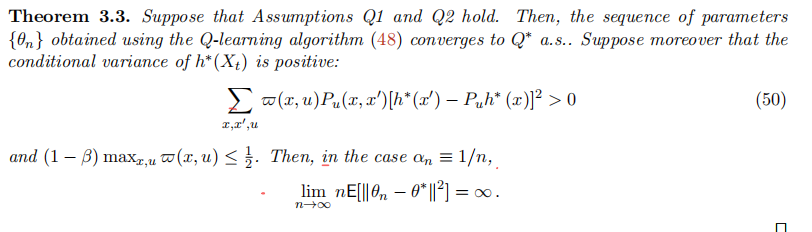
# 1**. Introduction**

Finding the optimal policy is always an essential question. However, suppose we don't know the environment. In that case, our learning control algorithms face a dilemma: they seek to learn action values from subsequent optimal behavior because they first need to behave poorly to explore all actions and hence find the optimal ones. "Off-policy learning," a set of methods consisting of target policy and behavior policy, is helping to solve the dilemma. As a standard "off-policy learning method," Watkins' Q-learning [1] method is an excellent way to approximate the optimal policy according to central limit theory [2]. However, Watkins' Q-learning algorithm suffers from slow convergence [3]. In order to solve this problem, many papers have appeared with proposed improvements, such as [4]. In our project, we first study a new learning algorithm, Zap Q-Learning [2], which can significantly improve the convergence rate without influencing the result. Next, we simulate Zap Q-learning with a neural network under the Cartpole problem [5] and compare its result with our baseline model Deep-Q network. Our final target is playing Atari 2600 computer games [6] with the Zap Q-Learning algorithm and comparing its performance with the Deep-Q network.

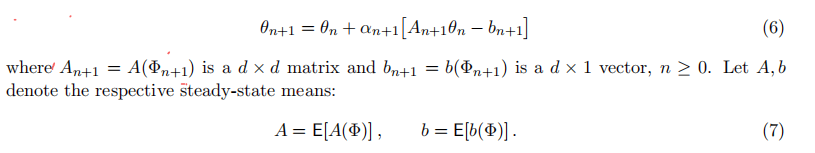
# **2. Methodology**

## **2.1 Zap-Q Learning**

When we use stochastic approximation algorithms to get our result, our update equation can always be expressed as *θn*+1 = *θn* + *αn*[*f*(*θn*) + ∆*n*+1] *, n ≥* 0 *,* where *θn* represents target sequence, like action value function in general Watkin's-Q learning algorithm [6]. According to Large Deviations or Central Limit Theorem (CLT) [1], our typical stochastic approximation algorithm above should take the following form:  will converges in distribution to Gaussian distribution , where Since truth covariance matrix is usual unknown, we usually use the scaled covariance to approximate the target, which looks like We can find that our result of Q-learning algorithms can arrive to the optimal value, however, asymptotic covariance will have problem for Q-learning algorithms according to the following theorem 3.3 in [2].



The fact that trace of Q-learning's asymptotic covariance will go to infinity lead many researchers to seek better methods. In many applications of reinforcement learning, we can actually make our stochastic approximation to be like the following form:

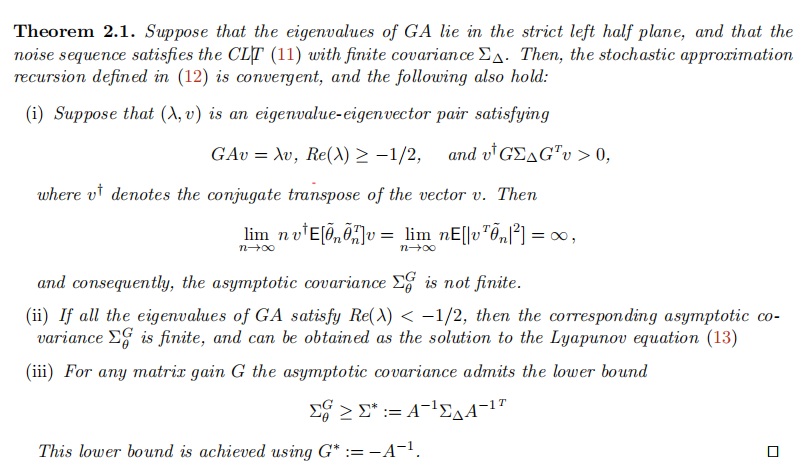


Where we assume  is the stationary realization of the Markov chain in this linear system. Since this sequence converges with probability to , we only to solve this classic Lyapunov equation to get the asymptotic covariance 

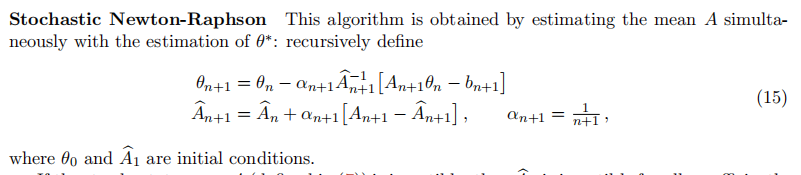
However, in order to make sure that the above equation has finite solution, matrix A should be Hurwitz, which means that all of the eigenvalues of A have real part that is strictly less than –0.5. In order to satisfy the requirement of matrix A, matrix gain method [2] is introduced, which take the following form.



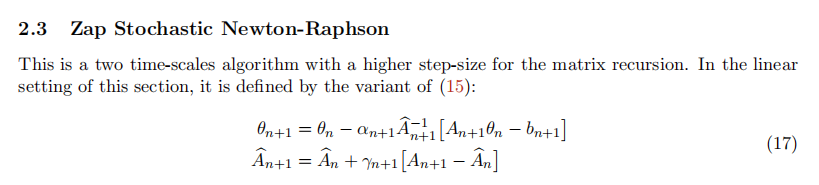
Furthermore, when ，asymptotic covariance can reach its lowest value, which like  Following theorem 2.1 in [2] summarize the convergence situation of this matrix gain stochastic algorithm and the optimal choice of matrix G.



At this point, stochastic Newton-Raphson (SNR) [2] is introduced.



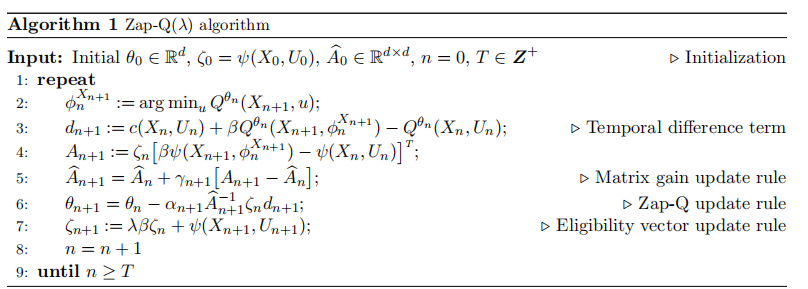
Facing that same step size is used in SNR, paper [2] introduces two time-scales stochastic approximation algorithms with matrix gain, called Zap Stochastic Newton-Raphson [2].



Furthermore, two step size sequence should satisfy the following requirements.

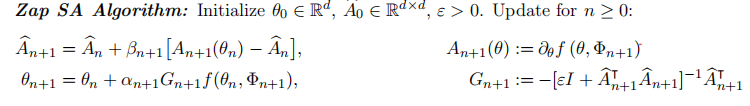
 where 

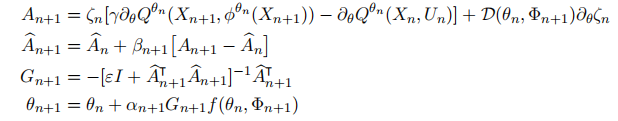
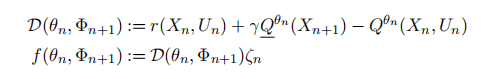
After combining with traditional Watkins’ Q-learning algorithm, here is the final pseudocode of Zap-Q learning algorithm in [2].



Where we usually choose and which represents the parameter and basic function used in Watkins’ Q learning algorithm.

Besides linear system, in fact paper [7] introduced a generalized Zap Q-learning method whose convergence theory is obtained even in a nonlinear function approximation setting. The main differences exist in the way to approximate matrix gain matrix G and A. Details of this algorithm are shown below.

After combined with Watkins’ Q- learning algorithms, we can get:

In our Cartpole simulation task, we use the generalized Zap Q- learning algorithms with neural networks.

## **2.2. DQN**

DQN (Deep Q-Network) is the first algorithm of Deep Reinforcement Learning (DRL), which introduces deep learning into reinforcement learning [9][10]. It has been demonstrated that this algorithm has a strong ability to solve complicated RL tasks, especially those which contain images. DQN has been widely used as the benchmark when researchers test new algorithms. In our project, we also use DQN as our benchmark.

## **2.3. Test environment**

Reinforcement learning (RL) is the branch of machine learning that is concerned with making sequences of decisions [11]. In the latest decade, RL has developed very quickly and numerous new algorithms, e.g., policy gradients and Q-learning, achieve good performance on difficult problems. Based on the rapid growth of RL, researchers need good benchmarks to test and compare different algorithms. To satisfy the demand, Brockman et al. develop a toolkit, OpenAI Gym(gym.openai.com), for reinforcement learning research [12]. It contains a collection of environments. Small-scale tasks from the RL literature, classic Atari games [13], Board games etc. are included. OpenAI Gym focuses on the constructure of environments and sample complexity. For users, it provides a user-friendly interface and helps with documentation.

In our project, we use Cartpole, a classic RL task, and Atari games, for instance Breakout, to test Zap Q-Learning and compare results from Zap Q-Learning and Deep Q-Learning.

交通信号灯

描述已自动生成电脑截图

中度可信度描述已自动生成

Figure (left) Cartpole. (breakout) Breakout

# **3. Results**

The Zap Q-learning algorithm was applied on two examples from OpenAI gym: Cartpole and Breakout. While Cartpole’s observation space only has 4 parameters, Breakout’s observation space is (210,160,3). Therefore, we chose different networks for the games: fully connected (FC) layer network for Cartpole and convolutional neural network (CNN)+FC for Breakout.

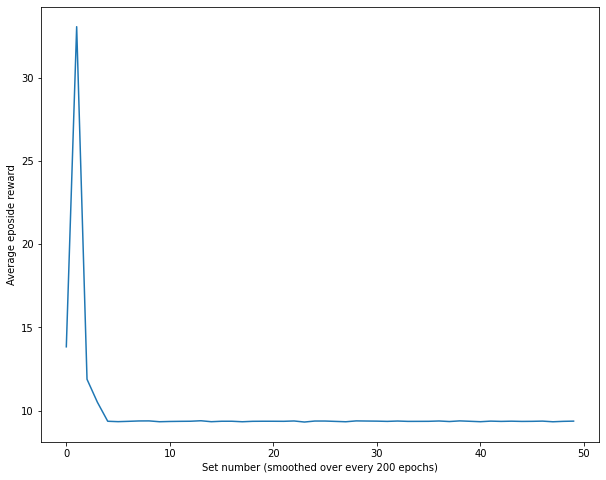
Originally, we planned to use the same network structure for a given game and apply DQN and Zap Q-learning. We succeeded in training DQN models on several Atari games, but a complication arose when we tried to run Zap Q-learning: we did not have enough memory. For training Zap Q-learning models, we used a 32 GB RAM virtual machine on the Google Cloud Platform, but it was still not enough to run the algorithm. Specifically, the program crashed when it tried to allocate memory for , , and b matrices. Each is a two-dimensional matrix with each dimension equal to the number of parameters of the whole neural network. The CNN + FC network has a total number of parameters 1684641, so the dimensions for those matrices are 1684641×1684641. The large number of parameters is due to the complexity of Atari game and, as a result, the need to use convolutional layers instead of FC layers. Simple games such as cartpole don’t have image as state; its observation is merely a one-dimensional four-element vector. However, for Atari games, even after down sampling the image, the observation space is still as large as 84\*84\*3, which can only be processed by convolutional layer effectively.

In order to run Zap Q-learning within the memory limitation, we reduced our model to a network consisting of 2 convolutional and 1 FC layers, with 27234 parameters in total. In comparison, the ANN for cartpole is of parameters 725. Implementation details are in the appendix.

Chart, histogram

Description automatically generated

Figure



Figure

Figure 2 and Figure 3 show the results of Cartpole using DQN and Zap Q-learning, respectively. Surprisingly, Zap Q-learning did not perform better; it did not even perform the same as in Chen’s work [7].

Chart

Description automatically generated

Figure

Shape

Description automatically generated with medium confidence

Figure

Figure 4 and Figure 5 show the results of Breakout using DQN and Zap Q-learning, respectively. DQN outperformed Zap Q-learning by a large margin.

# **4. Conclusion**

To start, we applied Zap Q-learning on Cartpole and attempted to reproduce the result in Chen’s paper [7]. Chen’s paper presented implementation details about the ‘meta-parameters in experiments. We used the same network architecture and hyperparameters as those in the paper, only to be disappointed by the result. Zap Q-learning model did not perform well on the breakout either. We also trained DQN models on some other games, but due to limited time, we did not finish training Zap Q-learning models on those games to compare with.

References

[1]. Watkins, C.J.C.H., Dayan, P., "Q-learning," Mach Learn, no. 8, pp. 279-292, 1992.

[2]. Adithya M. Devraj, Sean P. Meyn , Fastest Convergence for Q-Learning, 2018

[3]. C. Szepesvari. The asymptotic convergence-rate if Q-learning. In Proceedings of the 10th International Conference on Neural Information Processing Systems, pages 1064-1070. MIT Press,1997

[4]. E. Even-Dar and Y.Mansour. Learning rates for Q-learning. Journal of Machine Learning Research.5(Dec):1-25,2003

[5]. OpenAi Gym website. Github.com/openai/gym/wiki.

[6]. Mnih, Volodymyr, Playing Atari with deep reinforcement learning,2013

[7]. Shuhuang Chen, Adithya M.Devraj, Fan Lu, Ana Busic, Sean Meayn, Zap Q-learning with Nonlinear Function Approximation, 2020

[8]. Adithya M.Devraj, Ana Busic, and Sean Meyn, Zap Q-Learning- A User’s Guide

[9]. V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra and M. Riedmiller, "Playing Atari with Deep Reinforcement Learning," NIPS, 2013.

[10]. Mnih, V., Kavukcuoglu, K., Silver, D. et al. Human-level control through deep reinforcement learning. Nature 518, 529–533 (2015).

[11]. Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. (2016). OpenAI Gym. ArXiv, abs/1606.01540.

[12]. Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. (2016). OpenAI Gym. ArXiv, abs/1606.01540.

[13]. M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The Arcade Learning Environment: An evaluation platform for general agents. J. Artif. Intell. Res., 47:253–279, 2013

# Appendix

## Training DQN and Zap Q-learning models

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Games applied | Structure in original plan | Structure in actual training | Number of parameters |
| DQN | Cartpole | 3 FC layers of size 24\*16\*10 | 3 FC layers of size 24\*16\*10 | 712 |
| Zap Q-learning | Cartpole | 3 FC layers of size 24\*16\*10 | 3 FC layers of size 24\*16\*10 | 725 |
| DQN | Breakout, Enduro, Pong, Qbert | 3 Conv layers and 2 FC layers | 3 Conv layers and 2 FC layers | 1684641 |
| Zap Q-learning | Breakout | 3 Conv layers and 2 FC layers | 2 Conv layers and 1 FC layers | 27234 |

Table

In this project, we trained DQN models for the following artari games: Cartpole, Breakout, Enduro, Pong, Qbert. We trained Zap Q-learning model for Cartpole and Breakout. For Cartpole, we used the same network architecture and hyperparameters as in Chen’s paper[7]. The architectures are summarized in the table above.

Training DQN models was done on a 11GB RAM, NVIDIA T4 GPU virtual machine on the Google Cloud Platform. Training DQN for games other than CartPole took approximately 20 hours. Training Zap Q-learning model was done on a 32GB RAM virtual machine on the Google Cloud Platform. Training Zap Q-learning model for Cartpole took less than 1 hour and for Breakout approximately 48 hours.

Results of DQN model for Enduro, Pong and Qbert are shown here. Unfortunately we did not finish training Zap Q-learning models to compare with.

A picture containing chart

Description automatically generated

Figure Result of DQN on Enduro

A picture containing chart

Description automatically generated

Figure Result of DQN on Pong

Chart

Description automatically generated

Figure Result of DQN on Qbert

Hyperparameters can be found in the code submitted.