Reinforcement Learning Algorithm for Blackjack

Minheng Chen[1], Weiqi Zhang[1], Xiaokun Yue[2], and Zhirun Zhang[3]

[1] School of Computer Science and Engineering
Southeast University
Nanjing, China
[2] School of Software Engineering
Southeast University
Nanjing, China
[3] School of Artificial Intelligence
Southeast University
Nanjing, China

Abstract. We present a learning-based network for Blackjack by using reinforcement learning technique. The Deep Q-network follows the previous training method. And the proposed method get 4th place in a in-class competition.

1 Introduction

Blackjack is a globally popular banking game known as Twenty-One. The objective is to beat the dealer by reaching a higher score than the dealer without exceeding 21. In many U.S. casinos, players are limited to playing one to three positions at a table.The object of the game is to win money by creating card totals higher than those of the dealer's hand but not exceeding 21, or by stopping at a total in the hope that the dealer will bust. On their turn, players choose to "hit" (take a card), "stand" (end their turn and stop without taking a card), "double" (double their wager, take a single card, and finish), "split" (if the two cards have the same value, separate them to make two hands), or "surrender" (give up a half-bet and retire from the game). Number cards count as their number, the jack, queen, and king ("face cards" or "pictures") count as 10, and aces count as either 1 or 11 according to the player's choice. If the total exceeds 21 points, it busts, and all bets on it immediately lose. A player total of 21 on the first two cards is a "natural" or "blackjack", and the player wins immediately unless the dealer also has one, in which case the hand ties. In the case of a tie ("push" or "standoff"), bets are returned without adjustment. A blackjack beats any hand that is not a blackjack, even one with a value of 21.

Recently, with the development of machine learning and reinforcement learning techniques, Blackjack is now considered to be a ideal environment for developing learning-based algorithm.

In this work, we implement a simple version of Blackjack. In each round, the player only has two options: "hit" which will take a card, and 'stand' which end the turn. The player will "bust" if his hands exceed 21 points. After the player completes his hands (chooses “stand” and has not busted), the dealer then reals his hidden card and “hit” until obtaining at least 17 points.

In this paper, we proposed an algorithm for Blackjack: a Deep Q-Network(DQN), which shows great potentialities in this simple but interesting game.

2 Method and Experiment Settings

2.1 State Representation and Action Encoding

In this environment, we encode the state as an array [*player_score*, *dealer_score*, *usable*] where player score is the score currently obtained by the player, and the dealer score is derived from the card that faces up from the dealer. *usable* implies whether the player holds an Ace or not. There are two actions in this simple Blackjack. They are encoded as follows:

Action ID	Action
0	stand
1	hit

2.2 Deep Q-Network

DQN (Deep Q-Network) is a deep reinforcement learning algorithm proposed by researchers at DeepMind [1, 2]. DQN uses a deep neural network to learn the action-value function (Q-function) and maximizes the Q-function to determine the next action to take. DQN uses techniques such as experience replay and target networks to improve learning efficiency and stability. DQN has achieved success in many fields, including game AI, robot control, and autonomous driving.

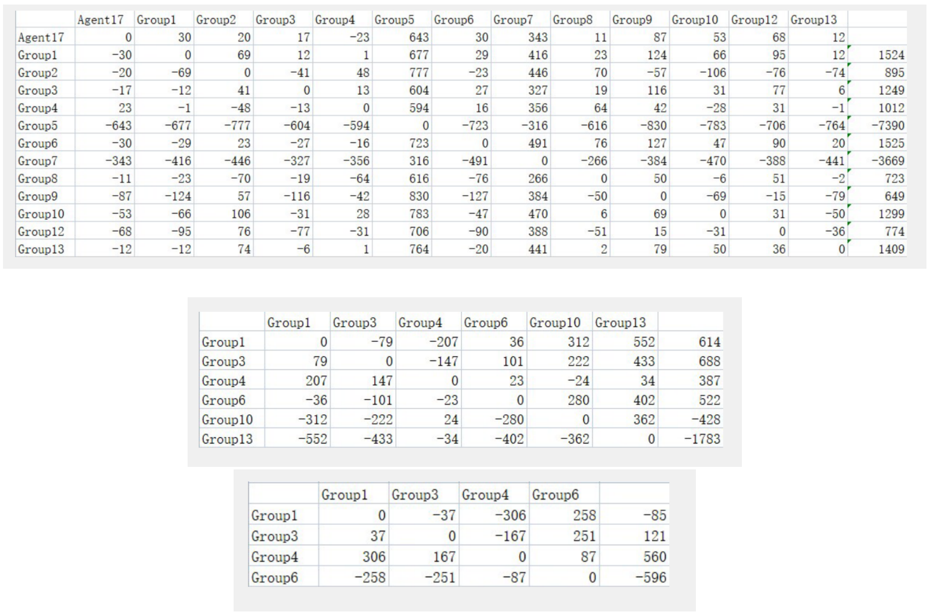
The operation principle of DQN is to use a deep neural network to approximate the Q-function, which predicts the expected future rewards for each possible action in a given state. DQN uses an iterative process to update the Q-function based on the difference between the predicted and actual rewards obtained from experience. The experience is stored in a replay buffer, and a random sample of experience is used to update the Q-function during each iteration. The target Q-value used for updating the Q-function is calculated using a separate target network to stabilize the learning process. The DQN algorithm maximizes the Q-function to determine the optimal action to take in a given state, and the process is repeated for each state encountered during training.

We implement the training environment by using [3]. We set the number of training episode as 10k and the number of games as 2000. And the learning rate 1e-5. The estimator network is constructed by three linear layers with dimension size [64,128,64], we use LeakyReLU as the activation function between each layer.

In the training environment, we set an agent as the dealer and our method as the player. The dealer follows the simplest policy: 'stand' when its points is greater than 17 else keeps 'hit'. We set our method as the player policy and use the simple policy mentioned above as our agent's dealer policy. We make a validation in the competition environment against with agent17, and find that we have a much higher winning percentage.

3 Experiment

The experiment is a in-class competition. Thirteen teams of students play Blackjack by using their own method. After a competitive process, in the end we got fourth place. The results are shown in Fig.1.



	Agent17	Group1	Group2	Group3	Group4	Group5	Group6	Group7	Group8	Group9	Group10	Group12	Group13	
Agent17	0	30	20	17	−23	643	30	343	11	87	53	68	12	
Group1	−30	0	69	12	1	677	29	416	23	124	66	95	12	1524
Group2	−20	−69	0	−41	48	777	−23	446	70	−57	−106	−76	−74	895
Group3	−17	−12	41	0	13	604	27	327	19	116	31	77	6	1249
Group4	23	−1	−48	−13	0	594	16	356	64	42	−28	31	−1	1012
Group5	−643	−677	−777	−604	−594	0	−723	−316	−616	−830	−783	−706	−764	−7390
Group6	−30	−29	23	−27	−16	723	0	491	76	127	47	90	20	1525
Group7	−343	−416	−446	−327	−356	316	−491	0	−266	−384	−470	−388	−441	−3669
Group8	−11	−23	−70	−19	−64	616	−76	266	0	50	−6	51	−2	723
Group9	−87	−124	57	−116	−42	830	−127	384	−50	0	−69	−15	−79	649
Group10	−53	−66	106	−31	28	783	−47	470	6	69	0	31	−50	1299
Group12	−68	−95	76	−77	−31	706	−90	388	−51	15	−31	0	−36	774
Group13	−12	−12	74	−6	1	764	−20	441	2	79	50	36	0	1409

	Group1	Group3	Group4	Group6	Group10	Group13	
Group1	0	−79	−207	36	312	552	614
Group3	79	0	−147	101	222	433	688
Group4	207	147	0	23	−24	34	387
Group6	−36	−101	−23	0	280	402	522
Group10	−312	−222	24	−280	0	362	−428
Group13	−552	−433	−34	−402	−362	0	−1783

	Group1	Group3	Group4	Group6	
Group1	0	−37	−306	258	−85
Group3	37	0	−167	251	121
Group4	306	167	0	87	560
Group6	−258	−251	−87	0	−596

Fig. 1: The result of the Blackjack competition, and our group number is 6.

4 Conclusion

In this paper, we proposed a reinforcement learning algorithm for Blackjack, and we got the fourth place in the in-class competition.

Contribution List.

Minheng Chen: The implementation and training of DQN; report writing.

Weiqi Zhang: Q-laerning method (failed).

Xiaokun Yue: Monte Carlo method (failed).

Zhirun Zhang: Data collection.

References

1. Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M.: Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602 (2013)
2. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., et al.: Human-level control through deep reinforcement learning. nature **518**(7540), 529–533 (2015)
3. Zha, D., Lai, K.H., Huang, S., Cao, Y., Reddy, K., Vargas, J., Nguyen, A., Wei, R., Guo, J., Hu, X.: Rlcard: A platform for reinforcement learning in card games. In: IJCAI (2020)