Baden-Wuerttemberg Cooperative State University Mannheim

Assignment

Reinforcement Learning

Course of Study Business Informatics

Field of Study Data Science

Author:	Valentin Moritz Müller
Matriculation Number:	4616344
Company:	SAP SE
Course:	WWI20DSB
Director of Studies	Prof. Dr. Bernhard Drabant
Academic Supervisor:	Janina Patzer
Completion Period:	22.05.2022 – 30.07.2023

Contents

List of Figures

List of Acronyms

A2C	Advantage Actor-Critic
A3C	Asynchronous Advantage Actor-Critic
DDDQN	Dueling Double Deep Q-Network
DDPG	Deep Deterministic Policy Gradient
DDQN	Double Deep Q-Network
DQN	Deep Q-Network
ER	Experience Replay
GAE	General Advantage Estimation
IID	Independent and Identically Distributed Data
NFQ	Neural Fitted Q Iteration
NN	(artificial) neural network
PER	Prioritized Experience Replay
PPO	Proximal Policy Optimization
RL	reinforcement learning
SAC	Soft Actor-Critic
TD	temporal difference
TD3	Twin-Delayed Deep Deterministic Policy Gradient
VPG	Vanilla Policy Gradient

1 Timeline of Deep Reinforcement Learning

REINFORCE

May 1992

The REINFORCE algorithm calculates the gradient of the predicted cumulative reward with respect to the policy parameters using the likelihood ratio trick: It multiplies the cumulative reward with the gradient of the logarithm of the policy function, which can be calculated using Monte Carlo techniques. It has good performance and convergence, guarantees for simple problems but suffers from high variance (Williams, 1992)

Vanilla Policy Gradient (VPG)

May 1992 or 1999 to 2000

The vanilla policy gradient or REINFORCE with baseline algorithm uses two networks: One for learning the policy, one for learning the state-value function. It uses the latter to then estimate the advantage function. It makes use of previous slides' loss function (Williams, 1992; Sutton et al., 1999).

Experience Replay (ER)

1992 to January 1993

ER tackles the correlation issue between samples by introducing a data structure called replay buffer or replay memory. It holds past experiences for several steps and enables the sampling of mini-batches from a diverse set of experiences. This results in lower-variance updates, better representativeness and stability as an Independent and Identically Distributed Data (IID)-like dataset (Lin, 1992).

Neural Fitted Q Iteration (NFQ)

October 2005

The algorithm NFQ consists of three main steps: (1) Sample experience tuples, (2) calculate off-policy targets, (3) fit Q-Function with MSE and RMSProp. The approach of NFQ is similar to a DQN, but fits the Q-function directly (Riedmiller, 2005).

Deep Q-Network (DQN)

December 2013

Many optimization methods were developed based on the IID distribution, which is not given in online learning. The DQN-approach tackles this problem by introducing a separate network for target updates, the target network. This stabilizes training, but slows down learning. The Q-values of the state-action pairs are the target of the model's estimation. DQNs utilize ER and fixed Q-targets (Mnih et al., 2013).

Prioritized Experience Replay (PER)

2015 to February 2016

Goal is to get agent to focus on important experiences. The fastest way to learn is by making mistakes, so "important" refers to experiences with a high discrepancy of expectation and reality, the difference between estimate and target value: Absolute temporal difference (TD) error.

PER Proportional prioritization: TD errors are noisy and shrink slowly in highly stochastic environments. We don‘t want noise to cause us to ignore experiences initially valued as zero, get stuck with noisy experiences or fixate on experiences with initially high TD error. That‘s why we can‘t sample greedily based on the TD error, but can calculate priorities based on it for proportional prioritization. Proportional prioritization is sensitive to outliers as they would be sampled more often.

PER Rank-based prioritization: A rank-based approach sorts the samples in descending order by the absolute TD error and chooses a sample by its position, i.e. its rank.

PER Weighted importance sampling: In off-policy algorithms we have the behavioral and target policy, each having a distribution. When using the calculated probabilities to sample experiences, we need to make sure to reflect the target distribution to prevent prioritization bias. Weighted importance sampling tackles the mismatch in distributions by changing the magnitude of the updates (Schaul et al., 2015).

General Advantage Estimation (GAE)

2015 to May 2016

Generalized advantage estimation uses a λ-target analogy for advantages. This further reduces variance by adding bias. The GAE algorithm was developed to address challenges related to estimating advantages accurately and efficiently, especially in high-dimensional continuous control tasks. Advantages represent how much better or worse an action is compared to the average action taken under a policy, and they play a crucial role in policy gradient methods to guide the optimization of the policy (Schulman et al., 2015).

Deep Deterministic Policy Gradient (DDPG)
2015 to May 2016

DDPG trains a policy approximating the optimal action, i.e. using a target deterministic policy function, instead of using the argmax on the Q-function like in DQN. To approximate a policy, a policy network which is differentiable with respect to the action is needed, and so is a continuous action space. The output is an action that maximizes the expected Q-value. DDPG utilizes the Actor-Critic-Architecture, Deterministic Policy Gradient, ER and Target Networks (Lillicrap et al., 2015).

Double Deep Q-Network (DDQN)
November 2015 to February 2017

Same as Q-Learning agents, DQN agents overestimate action-value functions by using the max-operator on estimates (positive bias) to combine DQN and double Q-Learning the algorithm is adapted, because otherwise there would be two Q-networks for both Q-functions and two corresponding target networks. In DDQN double learning is performed with the target network, but only the online network is trained, while cross-validation is performed with the target network (Wang et al., 2016; Van Hasselt et al., 2017).

Dueling Double Deep Q-Network (DDDQN)
November 2015 to February 2017

The DDQN can be optimized in its architecture: The dueling DDQN has two separate estimators for the state-value and action-advantage functions that have to share internal nodes. The two estimates are aggregated in one output, which contains the Q-Function estimate (Wang et al., 2016; Van Hasselt et al., 2017).

Asynchronous Advantage Actor-Critic (A3C)
June 2016

To further reduce variance, the asynchronous advantage actor-critic A3C algorithm uses n-step bootstrapping returns and concurrent agents to gather more experience samples. As on-policy methods can't reuse data generated by previous policies, multiple actor-workers can help by sampling data and asynchronously updating policy and value functions (Mnih et al., 2016).

Advantage Actor-Critic (A2C)

June 2016

This algorithm is a synchronous version of A3C. The key idea behind A2C is to use a single neural network to both approximate the policy (Actor) and estimate the value function (Critic). The Actor is responsible for selecting actions in the environment, while the Critic evaluates the value of states and actions to guide the learning process (Mnih et al., 2016).

Proximal Policy Optimization (PPO)

August 2017

Family of algorithms that alternate between sampling data through interaction with the environment, and optimizing a "surrogate" objective function using stochastic gradient ascent. Based on the same underlying architecture as A2C. Additionally introduces a surrogate objective function clipping the ratio between the new and old policies: This serves to prevent large policy updates, which in turn enables multiple optimization steps per mini-batch. Manages to strike a favorable balance between sample complexity, simplicity, and wall-time (Schulman et al., 2017).

Twin-Delayed Deep Deterministic Policy Gradient (TD3)

July 2018

TD3 was developed to address certain challenges and limitations faced by DDPG, particularly in terms of stability and the problem of overestimation bias. The key innovations and improvements introduced by TD3 are as follows: Clipped Double Q-Learning, Delayed Policy Updates, Target Policy Smoothing and Automatic Entropy Tuning (Fujimoto et al., 2018).

Soft Actor-Critic (SAC)

July 2018

Belongs to the off-policy algorithms,learns a stochastic policy. Utilized to solve the challenges of very high sample complexity and brittle convergence properties other model-free deep reinforcement learning (RL) methods suffer from. The objective is maximizing the long-term reward and entropy. Our entropy coefficient α is tuned automatically: It adjusts the value of alpha in proportion to the gradient of the expected entropy with respect to α. If the entropy of a policy is low, the update rule will increase α, which encourages more exploration, hence the name, soft (Haarnoja et al., 2018).

Note to the timeline

For many algorithms, an exact time of origin could not be determined, even with the corresponding paper. Others were introduced on the same conference, i.e. SAC and TD3.

2 Hands-on Project

2.1 Choice and Description of the Reinforcement Learning Environment

The environment that is utilized for the project is the Blackjack environment provided by the Python library *Gymnasium*. Initially, the usage of a more complex environment like Tetris or another one of the Atari 2600 environments was intended, but could not be implemented due to either the agent not working at all with no tutorials specific to *Gymnasium* available, their documentation lacking examples for Atari environments, and also LLMs in their free version unable to provide help since their current information (up to September 2021 for *ChatGPT*) predates *Gymnasium*, which was initially released in September 2022 (Farama, 2023; OpenAI, 2023). Blackjack was then chosen due to time constraints after failed attempts, as a simple environment allows for faster training and less complex agents, is a game well-known to the author and matches their interest in card games.

The environment has a discrete, basically boolean action space, which allows to either *Hit*, represented by 1 and meaning to take another card, or *Stick*, represented by 0 and meaning to not take another card. The observation space contains three discrete values, the first indicating the current sum, the second the value of the dealer's shown card, and the last one a flag indicting whether the player holds an ace, which can count as 11 or one depending on which is more useful. The player's current sum determines their actions. They can hit until they either choose to stick, or their total goes over 21, which is called a bust and results in an immediate loss. Once the player decides to stick, the dealer reveals their face-down card and draws additional cards until their total reaches 17 or higher. If the dealer busts, the player wins. If neither the player nor the dealer busts, the winner is determined by whose total is closer to 21, with a draw being possible if the sums of both players match (Farama, 2023).

2.2 Development of the Reinforcement Learning Agents

For this project, two agents are developed, some theoretical basics of which can be found in chapter 1. The first one is a Q-Learning learning agent, which utilizes a model-free algorithm focused on iteratively assigning a value to each action. For this implementation, the tutorial "Solving Blackjack with Q-Learning" from the documentation of *Gymnasium* was used as a basis, but required some fixing of bugs and extending with additional features, which was done with the help of *StackOverflow* and *ChatGPT* (Farama, 2023; OpenAI, 2023; Stack Overflow, 2023).

The second agent, which is intended for a comparison between the two approaches, is a the deep learning, model based approach of a DQN. The DQN is implemented with the deep learning framework *PyTorch*. DQNs extend the original Q-Learning algorithm with the usage of a (artificial) neural network (NN) to approximate the Q-Values. Normally, they are better suited for more complex tasks, which makes the comparison of both approaches interesting on a problem as simple as blackjack (PyTorch, 2023; Farama, 2023).

For both techniques, the same basic steps have to be implemented. The first is the creation of the agent itself, which only requires some basic setup for the Q-Learning agent, but more NN or model-typical attributes like a loss function and optimizer. Both agents require a dictionary with the Q-values, a function to enact an action, train the model or update the values, and decay the epsilon to manage exploration vs. exploitation. Both models then require a training loop, and are trained for 100,000 runs or epochs (Farama, 2023; PyTorch, 2023).

2.3 Results and Comparison

In the figures 2.1 till 2.4 below, the results of the training can be seen. When comparing the metrics in Figure 2.1 and Figure 2.3, they look similar, except for the error, which uses different metrics (with the one from the Q-Learning example weirdly even producing negative errors that increase with training due to its error function), so those are not comparable anyways. While the rewards are similar, the DQN takes slightly longer episodes, and the difference becomes clearer when having a look at the policies, here shown in Figure 2.2 and Figure 2.4 exemplary for the case an ace is available to the player. The

DQN is more aggressive, and is not lacking any random values in between like in the case the dealer shows an ace and the player has a sum of 12 or 17, with the former in particular not being clear when trying to find an explanation based on a combination of the rules of the game and other parts of the Q-Learning agents policy (Farama, 2023).

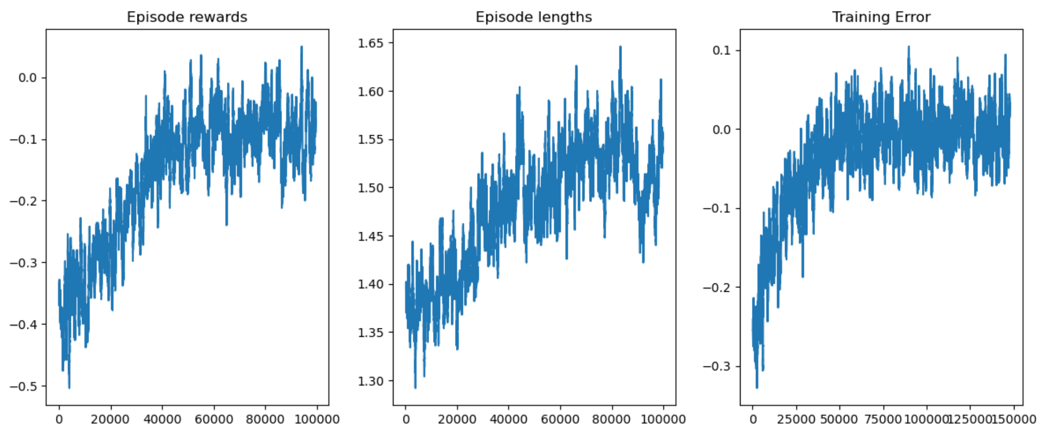

Figure 2.1: Performance of the Q-Learning agent
Source: Own illustration.

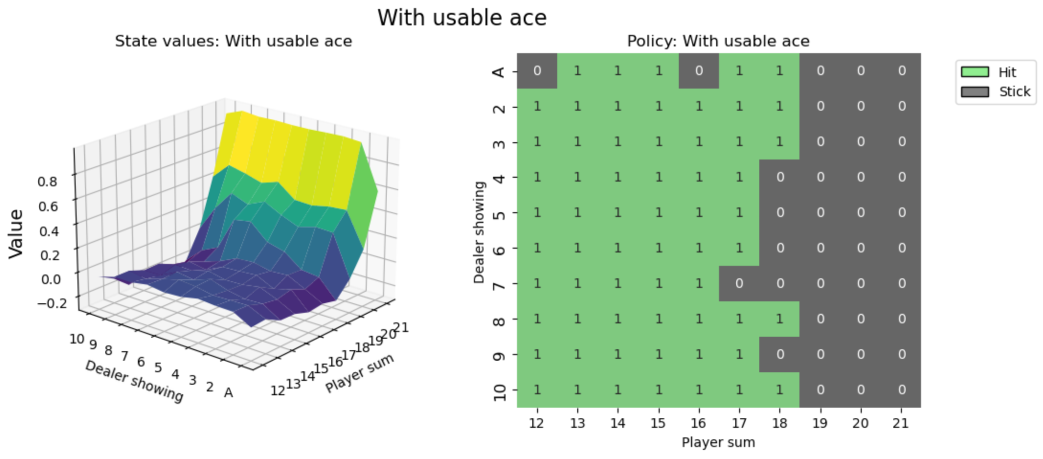

Figure 2.2: Policy of the Q-Learning agent when an ace is available
Source: Own illustration.

This shows that while the DQN has a slight edge, it is not able to clearly outperform the Q-Learning while being much more complex and costly regarding the training times, and thus not strictly better on this relatively simple problem. Regarding the performance in general it should be noted that Blackjack, as a game commonly played in casinos, does generally favour the dealer and thus it is not surprising that the episode rewards do generally not even reach zero, but are able to get close. Longer training was attempted, but did not

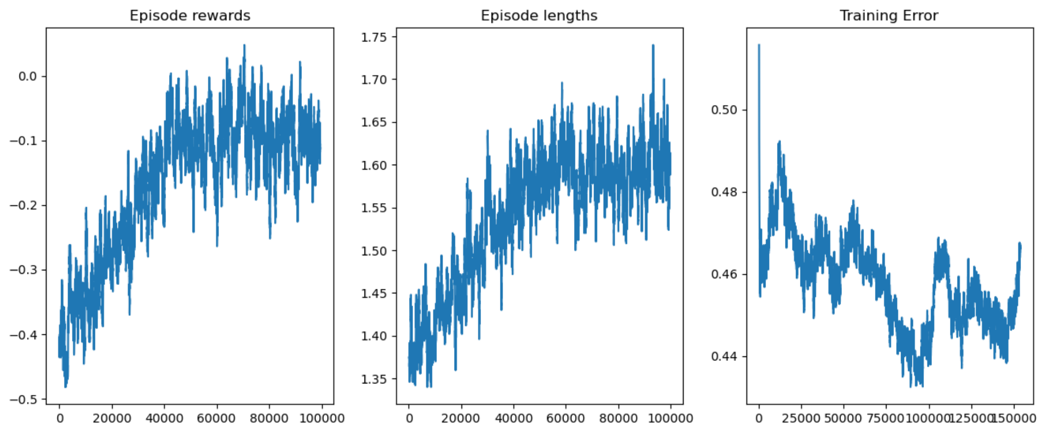

Figure 2.3: Performance of the DQN agent
Source: Own illustration.

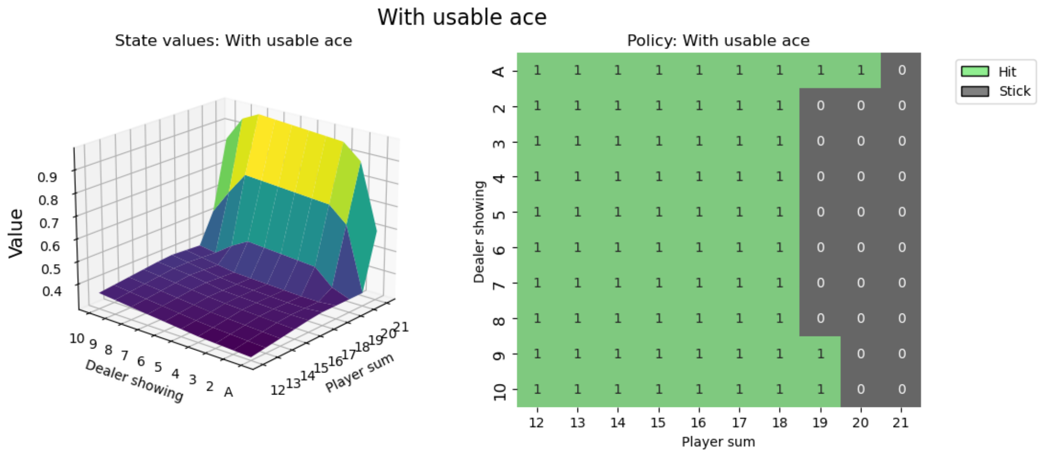

Figure 2.4: Policy of the DQN agent when an ace is available
Source: Own illustration.

lead to any notable increase, so this performance seems to be the cap, at least for agents based on Q-Learning or a technique building on it.

Bibliography

Borgonovo, E. (2017). *Sensitivity analysis: An introduction for the management scientist* (Vol. 251). Springer International Publishing.

Farama. (2023). Gymnasium documentation. Retrieved July 30, 2023, from https://gymnasium.farama.org/

Fujimoto, S., Hoof, H., & Meger, D. (2018). Addressing function approximation error in actor-critic methods. *International conference on machine learning*, 1587–1596.

Guyon, I., von Luxburg, U., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., & Garnett, R. (Eds.). (2017). *Advances in neural information processing systems*. Curran Associates, Inc.

Haarnoja, T., Zhou, A., Abbeel, P., & Levine, S. (2018). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *International conference on machine learning*, 1861–1870.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., & Wierstra, D. (2015). Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*.

Lin, L.-J. (1992). *Reinforcement learning for robots using neural networks*. Carnegie Mellon University.

Lundberg, S. M., & Lee, S.-I. (2017). A unified approach to interpreting model predictions. In I. Guyon, U. von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, & R. Garnett (Eds.), *Advances in neural information processing systems*. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2017/file/8a20a8621978632d76c43dfd28b67767-Paper.pdf

Milani, S., Topin, N., Veloso, M., & Fang, F. (2022). A survey of explainable reinforcement learning.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., & Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning. *International conference on machine learning*, 1928–1937.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Molnar, C. (2022). *Interpretable machine learning: A guide for making black box models explainable* (2nd ed.). https://christophm.github.io/interpretable-ml-book

OpenAI. (2023). Chatgpt: Get instant answers, find inspiration, learn something new. Retrieved July 30, 2023, from https://chat.openai.com/c/45592c3e-6e06-435a-99b8-a6e9aa330f48

Owen, G. (2013). *Game theory*. Emerald Group Publishing.

PyTorch. (2023). Pytorch documentation. Retrieved July 30, 2023, from https://pytorch.org/

Riedmiller, M. (2005). Neural fitted q iteration–first experiences with a data efficient neural reinforcement learning method. *Machine Learning: ECML 2005: 16th European Conference on Machine Learning, Porto, Portugal, October 3-7, 2005. Proceedings 16*, 317–328.

Schaul, T., Quan, J., Antonoglou, I., & Silver, D. (2015). Prioritized experience replay. *arXiv preprint arXiv:1511.05952*.

Schulman, J., Moritz, P., Levine, S., Jordan, M., & Abbeel, P. (2015). High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Shapley, L. S. (1953). A value for n-person games.

Sobol', I. M. (2001). Global sensitivity indices for nonlinear mathematical models and their monte carlo estimates. *Mathematics and Computers in Simulation*, *55*(1-3), 271–280.

Stack Overflow. (2023). Stack overflow - where developers learn, share, & build careers. Retrieved July 30, 2023, from https://stackoverflow.com/

Sutton, R. S., McAllester, D., Singh, S., & Mansour, Y. (1999). Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, *12*.

Van Hasselt, H., Guez, A., & Silver, D. (2017). Deep reinforcement learning with double q-learning. *Proceedings of the AAAI conference on artificial intelligence*, *30*(1).

Wang, Z., Schaul, T., Hessel, M., Hasselt, H., Lanctot, M., & Freitas, N. (2016). Dueling network architectures for deep reinforcement learning. *International conference on machine learning*, 1995–2003.

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, *8*, 229–256.