Heidelberg University
Institute of Computer Science

Project report for the lecture Fundamentals of Machine Learning

Reinforcement Learning for Bomberman

`https://github.com/nilskre/bomberman_rl`

Team Member: Felix Hausberger, 3661293,
Applied Computer Science
eb260@stud.uni-heidelberg.de

Team Member: Nils Krehl, 3664130,
Applied Computer Science
pu268@stud.uni-heidelberg.de

Abstract

Plagiarism statement

We certify that this report is our own work, based on our personal study and/or research and that we have acknowledged all material and sources used in its preparation, whether they be books, articles, reports, lecture notes, and any other kind of document, electronic or personal communication.

We also certify that this report has not previously been submitted for assessment in any other unit, except where specific permission has been granted from all unit coordinators involved, or at any other time in this unit, and that we have not copied in part or whole or otherwise plagiarized the work of other students and/or persons.

Contents

List of Abbreviations

DQN Deep-Q-Networks

MDP Markov Decision Process

1 Introduction

2 Fundamentals and Related Work

Q-Learning is a known off-policy and model-free approach to train an agent based on temporal difference in an environment that can be modeled as a Markov Decision Process (MDP). An agent therefore does not necessarily use the policy it is trained for and does not know the transition probabilities and rewards in the MDP beforehand. Equation 1 shows the iterative update formula for the Q-values that an online model uses to choose the right action [3].

$$Q_{k+1}(s,a) = (1-\alpha)Q_k(s,a) + \alpha(r + \gamma \max_{a'} Q_k(s',a')) \tag{1}$$

The problem with conventional Q-Learning is that in most of the cases the state dimension is far too high to explore and model the MDP entirely in foreseeable future. To deal with this problem the Q-values need to be approximated using a regression model. Deep neural networks have proven to be highly applicable for this task, which leads to the term of *Deep-Q-Learning* and respectively *Deep-Q-Networks* (DQN) for such network architectures. DQNs use the vectorized numerical state as its input and outputs the predicted Q-values. It learns through backpropagating the temporal difference error over each step for a single neuron. Note that for the Q-Learning algorithm a backpropagation is done after every step of the simulation. To avoid temporal correlation between succeeding experiences an experience buffer is used to randomly sample a training batch in each step. Also rare experiences will be used more frequently to update the model parameters using this approach. In the following papers regarding DQN architectures shall be introduced as well as papers dealing with the bomberman environment for reinforcement learning.

Paper [4] tackles the problem that the *max* operator in 1 often leads to overoptimistic value estimates as the DQN uses the same Q-values to both select and evaluate an action. It therefore changes the iterative update formula to 2.

$$Q_{k+1}(s,a) = (1-\alpha)Q_k(s,a) + \alpha(r + \gamma Q'_k(s', arg\max_{a'} Q_k(s',a'))) \tag{2}$$

Now a target DQN is used to separate the determination of the greedy policy, which is still done by the online network, from the Q-value estimation. Using this double DQN approach results in less overestimated Q-values and

therefore better policies by more accurate Q-value estimates. It also makes the learning process more stable and reliable. The weights are copied from the online network to the target network after a fixed amount of episodes.

Another optimization was introduced in paper [9]. It changes the architecture of the DQN by splitting it into two separate value streams. One stream estimates the state value function and the other one the state-dependent action advantage function. Both streams are then combined again using a special aggregating layer to produce an estimate of the Q-values (see Figure 1).

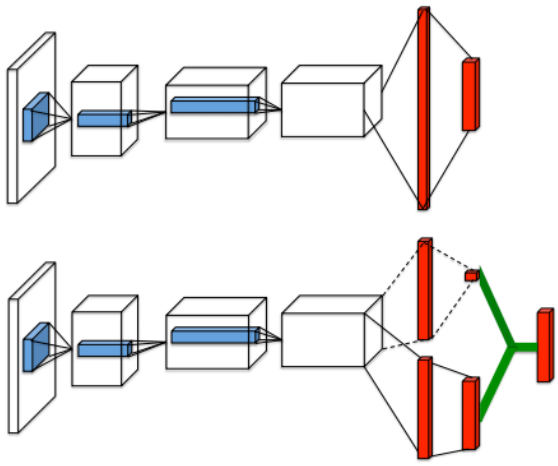

Figure 1: Architecture of a normal DQN (top) compared to a dueling DQN (bottom)

The state-dependent action advantage function is defined as

$$A^{\pi}(s,a) = A^{\pi}(s,a) - V^{\pi}(s) \tag{3}$$

and measures the importance of each action. The special aggregating layer uses Equation 4 to estimate the Q-values while also tackling the issue of identifiability.

$$Q(s,a;\theta,\alpha,\beta) = V(s;\theta,\beta) + (A(s,a;\theta,\alpha) - \frac{1}{|A|}\sum_{a'} A(s,a';\theta,\alpha)) \tag{4}$$

Using the dueling architecture approach an agent can learn which states are valuable independently of each action, which is esspecially useful in case actions do not influence the environment in any useful way. This leads to

a better and more robust policy evaluation in the presence of many similar-valued actions. Also when looking at the last layer of a conventional DQN (see Figure 1) it usually becomes much more sparse and biased. As the state value is modeled as a single neuron in the dueling architecture, learning the state value function becomes much more efficient.

Obviously, approaches exist that combine double Deep-Q-Learning with dueling architectures like [6].

Besides optimizing the learning process itself and the model architecture, [8] now proposes a way to sample more efficiently from the experience buffer. Each experience is assigned a learning priority score p_i with

$$p_i = \frac{1}{rank(i)} \tag{5}$$

where $rank(i)$ is the rank of experience i in the priority queue built upon the magnitude of the temporal difference error δ of each experience. The stochastic sampling probability of each experience is then calculated according to

$$P(i) = \frac{p_i^\alpha}{\sum_k p_k^\alpha} \quad with\ 0 \leq \alpha \leq 1. \tag{6}$$

The hyperparameter α determines the degree of using prioritization over random sampling. Using this stochastic sampling approach tackles the problem of a diversity loss and subsequent over-fitting when just greedyly sampling according to the magnitude of δ. One could also choose

$$p_i = |\delta_i| + \epsilon \tag{7}$$

instead of 5 but the latter is more prone to outliers. Furthermore, each gradient descent step during backpropagation needs to be weighted with

$$w_i = (\frac{1}{N}\frac{1}{P(i)})^\beta \tag{8}$$

to counter a bias towards high prioritized experiences introduced using the prioritized experience buffer. The weights should also be normalized with $\frac{1}{max_i w_i}$. As an unbiased nature of weight updates is esspecially important during the last training updates near convergence, β increases slowly from a start value β_0 to 1 over time.

There are several other improvements mentioned in the rainbow paper [5] like multi-step learning, distributional reinforcement learning and noisy nets, but these are out of the scope for this small research project.

Other papers were found that explicitly use DQNs within the Bomberman environment. One of them being [7] which introduces two novel exploration strategies, Error-Driven-ϵ and Interval-Q, and compares them to conventional exploration strategies like Diminishing ϵ-Greedy and Max-Boltzmann, whereas Max-Boltzmann with decreasing temperature parameter still performs best in the long run by empirical evaluation. Nevertheless Error-Driven-ϵ, despite being less stable, learns faster than all other exploration techniques. The paper also gives an approach to encode the state. Therefore, for each cell four values are computed:

- Free, breakable and obstructed cells are encoded as either 1, 0 or -1,
- The position of the player and opponent players are encoded as 1, free cells as 0 and
- The danger score for each cell is calculated as , which gets an additional negative sign in case a bomb was planted by the player itself.

The paper also gives valuable insights into the configuration of hyperparameters, rewards and the amount of training needed until convergence, which is about 100 generations Ã 10.000 episodes.

[2] uses an imitation-based learner that trains its model with the actor-critic proximal-policy optimization method in the Bomberman environment. Here insights about how rewards need to be chosen and which state representation to choose can also be derived.

Bomberman seems to provide a perfect environment to try out different reinforcement learning methods, which is why [1] provided an artificial intelligence platform around Bomberman including several associated intelligent agents and empirical experiments.

3 Approach

3.1 Reinforcement Learning Method and Regression Model

- Our model
- Our exploration method

3.2 Training process

- Experience buffer

4 Experimental results

5 Conclusion

A Appendix

A.1 Appendix A

References

[1] Manuel AntÃ³*niodaCruzLopes*. "Bomberman as an Artificial Intelligence Platform". PhD thesis. Departamento de CiÃªncia de Computadores: Universidade do Porto, 2016. URL: https://repositorio-aberto.up.pt/bitstream/10216/91011/2/176444.pdf.

[2] Ãcaro Goulart Faria Motta FranÃ§a, Aline Paes, and Esteban Clua. "Learning How to Play Bomberman with Deep Reinforcement and Imitation Learning". In: *Entertainment Computing and Serious Games* (2019). DOI: 10.1007/978-3-030-34644-7_10.

[3] AurÃ©lien GÃ©ron. *Praxiseinstieg Machine Learning mit Scikit-Learn und TensorFlow*. O'REILLEY, 2018. ISBN: 978-3-96006-061-8.

[4] Hado van Hasselt, Arthur Guez, and David Silver. "Deep Reinforcement Learning with Double Q-learning". In: *arXiv* (2015). URL: https://arxiv.org/abs/1509.06461.

[5] Matteo Hessel et al. "Rainbow: Combining Improvements in Deep Reinforcement Learning". In: *arXiv* (2017). URL: https://arxiv.org/abs/1710.02298.

[6] Ying Huang, GuoLiang Wei, and YongXiong Wang. "V-D D3QN: the Variant of Double Deep Q-Learning Network with Dueling Architecture". In: 2018. DOI: 10.23919/ChiCC.2018.8483478.

[7] Joseph Groot Kormelink, Madalina Drugan, and Marco Wiering. "Exploration Methods for Connectionist Q-Learning in Bomberman". In: 2018. DOI: 10.5220/0006556403550362.

[8] Tom Schaul et al. "Prioritized Experience Replay". In: *arXiv* (2016). URL: https://arxiv.org/abs/1511.05952.

[9] Ziyu Wang et al. "Dueling Network Architectures for Deep Reinforcement Learning". In: *arXiv* (2016). URL: https://arxiv.org/abs/1511.06581.