**Noname manuscript No.**
(will be inserted by the editor)

# Neural Networks and their Application in Reinforcement Learning

## Reinforcement Learning Seminar – Winter Semester 2018/19

**Fabian Otto**

Received: date / Accepted: date

**Abstract** Insert your abstract here. Include keywords, PACS and mathematical subject classification numbers as needed.

**Keywords** Reinforcement Learning · Neural Networks

## 1 Introduction

Give introduction with NN hype or similar things. As well as the importance. Define RL and the method in general. Show clear distinction to SL. Maybe include success of recent things in CV and NLP, might also be possible in **??** to transition to more recent approaches in RL. Introduce RL goal and notation, short intro ro MDPs.

## 2 Preliminaries

A Markov Decision Process (MDP) is defined as a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R})$, where $\mathcal{S}$ is a finite set of states, $\mathcal{A}$ is a finite set of actions, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \Pi(\mathcal{S})$ represents the state transition function, which returns a probability distribution over states and $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ returns the expected immediate reward for taking each action in each state. This paper mainly considers the infinite horizon MDP, which tries to maximize the expected discounted cumulative reward:

$$R_t = \mathbb{E}\left[\sum_{\tau=t}^{k=\infty} \gamma^{\tau-t} r_t\right]$$

Here $\gamma \in [0, 1)$ describes the discount factor, which trades of immediate and future rewards, and $r_t$ the received reward at step $t$. By choosing $\gamma = 1$ and setting the horizon $k$ to an finite value, the optimization target of finite horizon MDP can be

Fabian Otto
Technische Universität Darmstadt, Computer Science Department
E-mail: fabian.otto@stud.tu-darmstadt.de

described.
RL algorithms now seek a policy $\pi : \mathcal{S} \rightarrow \mathcal{A}$, which describes the optimal behavior to maximize the expected future reward from all states. Therefor, we define an action-value function, which returns the value of taking an action $a$ in state $s$, under the policy $\pi$ as:

$$Q_\pi(s,a) = \mathbb{E}\left[R_t | s_t = s, a_t = a, \pi\right] = \mathcal{R}(s, \pi(s)) + \gamma \sum_{s \in \mathcal{S}} \mathcal{T}(s,a,s') \max_{a'} Q_\pi(s',a')$$

In order to find the best policy, value-based methods are interested in finding the optimal action-value function $Q^*(s,a) = \max_\pi Q_\pi(s,a)$, which is equivalent to finding the optimal policy

$$\pi^*(s) = \arg\max_a \mathcal{R}(s,a) + \gamma \sum_{s \in \mathcal{S}} \mathcal{T}(s,a,s') \max_{a'} Q^*(s',a')$$

Alternatively, the optimal value function $V^*(s) = \mathbb{E}\left[R_t | s_t = s, \pi\right] = \max_a Q^*(s,a)$ combined with a model of the environment can be used.
In contrast to value-based methods, policy-based methods directly parameterize a policy $\pi(s;\theta)$. One such method is REINFORCE [33], it updates the policy parameters $\theta$ in the direction $\nabla_\theta \mathbb{E}[R] \approx \nabla_\theta \log \pi(s;\theta) R$. In order to reduce variance and keeping the estimate unbiased, a baseline $b(s)$ is subtracted from the return $\nabla_\theta \log \pi(s;\theta)(R - b(s))$. For this baseline, it is common to use an estimate of the value function $b(s) \approx V^\pi(s)$. Consequently, $R - b(s)$ can be seen as an estimate of the advantage function $A_\pi(s,a) = Q_\pi(s,a) - V_\pi(s)$.

## 3 Neural Networks and their history

(see Sect. **??**). [**?**] and [**?**]. Describe formally how NN are working, how can they be trained, what other methods do we have, etc. What did lead to the rise and fall of NNs throughout time. Create good transition to recent approaches.

## 4 Algorithms

This section provides an overview of NN based RL algorithms including earlier work and its improvements as well as current state-of-the-art. Most algorithms can be either classified as on-policy or off-policy algorithms. On-policy algorithms estimate the value of policy while using it for control whereas off-policy algorithms, the generating policy, called *behavior policy*, is not necessarily related to the *target policy*, which is improved and evaluated. This allows to use a deterministic target policy, while still sampling all actions through the behavior policy. [28, chapter 5]

### 4.1 Off-Policy

*MDPs* One of the earlier NN based off-policy algorithms, named Neural fitted Q Iteration (NFQ), was published by Riedmiller [20] in 2005. NFQ is a model-free algorithm, based on Q-Learning [32] and approximates the action-value function with multilayer-perceptrons. Compared to other approaches at the time, NFQ allows to learn with a relatively good data efficiency. Further, batches from a replay memory [15] are used in order to train the multilayer-perceptron with resilient backpropagation (RPROP) [21]. Thereby, it is possible to dynamically add new samples to the replay memory during learning. Motivated by this idea as well as the success of TD-gammon [29] the arguably biggest improvement in NN based RL – Deep Q-Networks (DQN) – were introduced by Mnih, et al. [19]. They built upon their prior results [18] on the Atari Arcade Environment [2] which already achieved state-of-the-art performance for some of the games. DQNs introduce better scalability to larger data sets by replacing NFQ's RPROP with Stochastic Gradient Descent (SGD) updates. Further, they allow training end-to-end from the raw visual input utilizing convolutional neural networks. Benchmarks of more recent version [19] show an improved performance as well as a better generalization to more Atari games compared to initial work [18]. The performance improvement was mainly achieved by introducing a second target Q-network, which is only periodically updated and thereby reduces sample correlations. However, van Hasselt, et al. [8] show that the current DQN [19] approach is not sufficient to avoid overestimations of action-values under certain conditions. This in itself is not harmful to the policy's performance, but if the overestimation is not uniform and not concentrated at the states of interest, it might affect the policy negatively. To mitigate the risk of overestimation, they propose Double DQN (DDQN). Double Q-Learning in general tries to decouple the selection and evaluation of actions by learning two action-value functions. One determines the greedy policy and the other the value of this policy. In order to reduce the computational cost, van Hasselt, et al. utilize the already existing online Q-network for determining the greedy policy and the target Q-network for estimating its value. Further, this allows them to achieve better performance on most of the Atari games. Decoupling is also done in a more extreme form with Dueling Double DQNs (DDDQN) from Wang, et al. [31]. They introduce two separate function approximators, one for the state-value function and one for the state dependent action advantage function, in order to represent the action-value[1]. These approximators allow the dueling architecture to learn about the value of a state regardless of the action taken. This is especially important as in many states, the action has no repercussions about the near future. Additional improvements, include gradient clipping and prioritized experience replay (PER) [22]. Combining PER with DDQN [22] showed that sampling rare experiences with a higher probability makes learning with replay memories more efficient [15]. However, sampling non-uniformly from the replay memory introduces a bias, that has to be corrected during the update step.

All previous approaches are using environments, which do not have to deal with delayed and sparse feedback. In real world applications this is often not the case and algorithms still need the ability to learn. One key problem, which arises during training, is insufficient exploration, thereby unstable policies. Using intrinsically motivated agents, exploration can be achieved for the agent itself rather for an external goal. This idea is implemented by Kulkarni, et al. [13] in hierarchical-DQN (h-DQN), which is based on a hierarchy of two DQNs. A top level *meta-controller* is selecting a subgoal in order to optimize the extrinsic reward. The lower level

[1] Both estimators share parameters in the networks's convolutional part and are trained together end-to-end.

*controller* maximizes an intrinsic reward by solving the subgoal. This allows h-DQN to achieve a significant better performance on the Atari game Montezuma's Revenge.

As the above results show, classical Q-learning [32] is quite successful for discrete action spaces, real world problems however often require continuous action spaces. Further, it is usually hard in real world applications to collect data from e.g. robots, and therefore dealing with high sample complexity. Normalized advantage functions (NAF) [5] approach both problems with a DQN based algorithm. Similar to DDDQN [31], the Q-network is represented by one state-value function output and one state dependent action advantage function. In order to reduce sample complexity, NAF incorporates *imagination rollouts*. Those synthetic on-policy samples are created by utilizing a mixture of iterative LQG (iLQG) [30] and on-policy trajectories for real world rollouts. Afterwards, the learned model for the states is used to generate replay memory entries. This can be seen as a scalable variant of Dyna-Q [27]. As a drawback, NAF failed to show improvements using iLQG compared to on-policy samples, however, iLQG might be desirable when damage avoidance is crucial. Lillicrap, et al. [14] also adapt the fundamental idea of DQN [19] and make it applicable to the continuous action domain. However, they do not follow a value based learning and introduce Deep Deterministic Policy Gradients (DDPG), a model-free Actor-Critic (AC) algorithm with approximate Q-Learning based on deterministic policy gradients (DPG) [26]. DQN's [19] architecture is improved by adding batch normalization layers [11] to deal with different physical units in the observation space. In order to stabilize policies, they adapt the idea of target networks from DQN [19] and use "soft" target updates for AC. Albeit, this stability is, according to Haaranoja, et al. [6], extremely difficult to achieve and further, DDPG is brittle to hyperparameter choices. As consequence, Haaranoja, et al. [6] present soft actor critic (SAC). They combine the AC approach from DDPG [14] with maximum entropy reinforcement learning, which enables exploration and stability. Similarly to NAF [5], they also approach the problem of sample efficiency and gain significant improvements compared DDPG and other on-policy methods. In general they also found modeling the state-value function and the action-value function with two separate NNs improved the stability of SAC. Another extension of DDPG [14], which was developed in parallel with SAC continues to addresses the hyperparameter sensitivity. Further, they show that overestimation of the action-value function is also a present issue for AC setting. DDQN [8] approach to this in the discrete case is not easily applicable to the continuous actions space, therefor Fujimoto, et al. [3] introduce twin delayed DDPG (TD3). In order to mitigate the above issues, they apply three key improvements. A clipped variant of double Q-Learning [7] trains two separate Q-Networks in order to reduce the overestimation bias. Policy smoothing, similar to expected SARSA [25], adds noise to the target action to avoid exploiting Q-function errors and brings action-values of similar actions closer together. The most important improvement, however, are delayed updates, i.e.the policy network as well as the target networks are updated less frequently than the Q-network. This avoids that the policy is optimized based on incorrect value estimates and is less likely to diverge. Building upon the idea of entropy reinforcement learning from SAC, Maximum a-posteriori policy optimization (MPO) [1] transforms the RL problem in an inference problem, which allows them to utilize expectation maximization (EM) for training, while optimizing a relative-entropy objective. Combining the distributed framework of RL algorithms

[17] as well as the distributional per on-policy as well as off-policy improvements from last years in one single algorithm

*POMDPs*

### 4.2 On-Policy

DDPG established a policy gradient based method, which works off-policy. However, must policy gradient methods are working on-policy. Comparably important to DQN [19] in the off-policy setting is the introduction of Asynchronous Advantage Actor Critic (A3C) [17] for the on-policy setting. Besides A3C, Mnih, et al. also introduce asynchronous off-policy methods for Q-Learning as well as SARSA. However, they show that A3C outperforms the off-policy methods. Utilizing multiple worker threads allows A3C to experience a larger space of the environment and incorporate this into one shared network. Consequently, the stability and robustness of the training process is increased without incorporating a replay memory. Further, less computational power is required to achieve better results compared to DQN [19].[2] Similar to SAC [6] entropy is employed, however only as regularizer and not as optimization target. A3C was also tested for continuous action spaces and was able to learn reliable policies. Following concept of policy gradients, Schulman, et al. [23] introduce trust region policy optimization (TRPO). They criticize that first order gradient methods are often overconfident and not accurate enough in curved areas, which results in losing already made learning progress. Consequently, TRPO applies the conjugate gradient method (CG) to the natural policy gradient [12]. The optimization is based on a form of minorization-maximization [10] and optimizes a surrogate function under a Kullback-Leibler (KL) constrained objective. This trust region constraint represents a pessimistic/lower bound on the performance of the policy and guarantees monotonic improvement locally around the current policy. In order to solve for the inverse of the fisher information matrix (FIM), which is required for the CG update, truncated natural policy gradients are applied to find an approximate solution. Before the update a backtracking line search is applied, which allows to assume larger trust regions and therefore larger update steps. However, one big drawback in practice is the sample efficiency of TRPO. Theoretically, TRPO can be applied to any policy, but, due to the lack of scalability, it is not practical to use deep neural networks policies. This shortcoming is addressed by Proximal Policy Optimization (PPO) [24]. PPO keeps all monotonic improvement guarantees while using first order optimization. This can be achieved two ways:

1. The KL constrained objective is replaced with a KL penalized objective and the corresponding penalty coefficient is automatically adapted each step.
2. The objective can be clipped, i.e. the probability ratio of old and new policy, which results of an lower bound for the unclipped objective.

Typically, clipping the objective shows better performance than the adaptive KL penalty. PPO is also combined with A3C's idea of asynchronicity and implemented

[2] A3C only utilizes CPU computation compared to DQN's GPU computation for the Atari environments.

in a distributed scenario as Distributed PPO (DPPO) [9]. Besides PPO, actor critic using kronecker-factored trust region (ACKTR)[34] also approaches TRPO's problem of sample complexity. ACKTR utlizes A3C's advantage function and approximates the natural gradient with kronecker-factored approximate curvature (K-FAC) [4] [16], which offers a comparable cost to SGD. By maintaining a running average K-FAC is able to reduce variance and achieve scalability, therefore ACKTR is more efficient in computing the inverse FIM. Further, ACKTR is not only able to increase the performance and sample efficiency compared to TRPO but also compared to the synchronous version of A3C - A2C [17].

### 4.3 Real World Applications

Talk about DeepMinds AlphaZero, TDGammon, Atari Game Systems, e.g. Minh But also applications outside of Games, maybe seperate this into two different subsections.

## References

1. Abdolmaleki, A., Springenberg, J.T., Tassa, Y., Munos, R., Heess, N., Riedmiller, M.: Maximum a Posteriori Policy Optimisation (2018). URL http://arxiv.org/abs/1806.06920
2. Bellemare, M.G., Naddaf, Y., Veness, J., Bowling, M.: The Arcade Learning Environment: An Evaluation Platform for General Agents. Journal of Artificial Intelligence Research **47**, 253–279 (2013). DOI 10.1613/jair.3912. URL http://arxiv.org/abs/1207.4708
3. Fujimoto, S., van Hoof, H., Meger, D.: Addressing Function Approximation Error in Actor-Critic Methods. In: International Conference for Machine Learning (2018). URL http://arxiv.org/abs/1802.09477
4. Grosse, R., Martens, J.: A Kronecker-factored Approximate Fisher Matrix for Convolution Layers. In: Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48, ICML'16, pp. 573–582. JMLR.org (2016). URL http://dl.acm.org/citation.cfm?id=3045390.3045452
5. Gu, S., Lillicrap, T., Sutskever, I., Levine, S.: Continuous Deep Q-Learning with Model-based Acceleration. In: ICML'16 Proceedings of the 33rd International Conference on International Conference onMachine Learning - Volume 48, pp. 2829–2838. JMLR.org, New York, NY, USA (2016). URL http://arxiv.org/abs/1603.00748
6. Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu, H., Gupta, A., Abbeel, P., Levine, S.: Soft Actor-Critic Algorithms and Applications (2018). URL http://arxiv.org/abs/1812.05905
7. van Hasselt, H.: Double Q-learning. In: J.D. Lafferty, C.K.I. Williams, J. Shawe-Taylor, R.S. Zemel, A. Culotta (eds.) Advances in Neural Information Processing Systems 23, pp. 2613–2621. Curran Associates, Inc. (2010). URL http://papers.nips.cc/paper/3964-double-q-learning.pdf
8. van Hasselt, H., Guez, A., Silver, D.: Deep Reinforcement Learning with Double Q-learning. In: AAAI Conference on Artificial Intelligence, pp. 2094–2100. Phoenix, Arizona (2016). URL http://arxiv.org/abs/1509.06461
9. Heess, N., TB, D., Sriram, S., Lemmon, J., Merel, J., Wayne, G., Tassa, Y., Erez, T., Wang, Z., Eslami, S.M.A., Riedmiller, M., Silver, D.: Emergence of Locomotion Behaviours in Rich Environments (2017). URL http://arxiv.org/abs/1707.02286
10. Hunter, D.R., Lange, K.: A tutorial on MM algorithms. The American Statistician **58**(1), 30–37 (2004)
11. Ioffe, S., Szegedy, C.: Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In: International Conference on Machine Learning (ICML), ICML'15, pp. 448–456. JMLR.org (2015). URL http://dl.acm.org/citation.cfm?id=3045118.3045167

12. Kakade, S.: A Natural Policy Gradient. In: Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic, NIPS'01, pp. 1531–1538. MIT Press, Cambridge, MA, USA (2001). URL http://dl.acm.org/citation.cfm?id=2980539.2980738
13. Kulkarni, T.D., Narasimhan, K., Saeedi, A., Tenenbaum, J.: Hierarchical Deep Reinforcement Learning: Integrating Temporal Abstraction and Intrinsic Motivation. In: D.D. Lee, M. Sugiyama, U.V. Luxburg, I. Guyon, R. Garnett (eds.) Advances in Neural Information Processing Systems 29, pp. 3675–3683. Curran Associates, Inc. (2016). URL http://papers.nips.cc/paper/6233-hierarchical-deep-reinforcement-learning-integrating-temporal-abstraction-and-intrinsic-motivation.pdf
14. Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D.: Continuous control with deep reinforcement learning. In: International Conference on Learning Representations (ICLR) 2016. London, UK (2016). URL http://arxiv.org/abs/1509.02971
15. Lin, L.J.: Self-improving reactive agents based on reinforcement learning, planning and teaching. Machine Learning **8**(3), 293–321 (1992). DOI 10.1007/BF00992699. URL https://doi.org/10.1007/BF00992699
16. Martens, J., Grosse, R.: Optimizing Neural Networks with Kronecker-factored Approximate Curvature (2015). URL http://arxiv.org/abs/1503.05671
17. Mnih, V., Badia, A.P.A.P., Mirza, M., Graves, A., Lillicrap, T.P., Harley, T., Silver, D., Kavukcuoglu, K.: Asynchronous Methods for Deep Reinforcement Learning. In: M.F. Balcan, K.Q. Weinberger (eds.) Proceedings of The 33rd International Conference on Machine Learning, *Proceedings of Machine Learning Research*, vol. 48, pp. 1928–1937. PMLR, New York, New York, USA (2016). URL http://proceedings.mlr.press/v48/mniha16.html http://arxiv.org/abs/1602.01783
18. Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M.: Playing Atari with Deep Reinforcement Learning. NIPS Deep Learning Workshop 2013 (2013). URL http://arxiv.org/abs/1312.5602
19. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., Hassabis, D.: Human-level control through deep reinforcement learning. Nature **518**(7540), 529–533 (2015). DOI 10.1038/nature14236. URL http://www.nature.com/articles/nature14236
20. Riedmiller, M.: Neural Fitted Q Iteration - First Experiences with a Data Efficient Neural Reinforcement Learning Method. In: J. Gama, R. Camacho, P.B. Brazdil, A.M. Jorge, L. Torgo (eds.) Machine Learning: ECML 2005, pp. 317–328. Springer Berlin Heidelberg, Berlin, Heidelberg (2005)
21. Riedmiller, M., Braun, H.: A Direct Adaptive Method for Faster Backpropagation Learning: The RPROP Algorithm. In: IEEE International Conference on Neural Networks (IJCNN), pp. 586–591 (1993)
22. Schaul, T., Quan, J., Antonoglou, I., Silver, D.: Prioritized Experience Replay. In: Interantional Conference for Learning Representations (ICLR) (2015). URL http://arxiv.org/abs/1511.05952
23. Schulman, J., Levine, S., Abbeel, P., Jordan, M., Moritz, P.: Trust Region Policy Optimization. In: F. Bach, D. Blei (eds.) Proc. 32nd Int. Conf. Mach. Learn., *Proceedings of Machine Learning Research*, vol. 37, pp. 1889–1897. PMLR, Lille, France (2015). URL http://proceedings.mlr.press/v37/schulman15.html
24. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal Policy Optimization Algorithms (2017). URL http://arxiv.org/abs/1707.06347
25. van Seijen, H., van Hasselt, H., Whiteson, S., Wiering, M.: A theoretical and empirical analysis of Expected Sarsa. In: 2009 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning, pp. 177–184 (2009). DOI 10.1109/ADPRL.2009.4927542
26. Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., Riedmiller, M.: Deterministic Policy Gradient Algorithms. In: International Conference on Machine Learning (ICML), ICML'14, pp. I–387—-I–395. JMLR.org, Beijing, China (2014). URL http://dl.acm.org/citation.cfm?id=3044805.3044850
27. Sutton, R.S.: Integrated architectures for learning, planning, reacting based on approxmiate dynmaic programming. In: International Conference for Machine Learning (ICML)2, pp. 216–224 (1990)
28. Sutton, R.S., Barto, A.G.: Reinforcement learning: An introduction, second edi edn. MIT Press, Cambridge, MA (2018). URL http://incompleteideas.net/book/the-book.html

29. Tesauro, G.: TD-Gammon, a Self-Teaching Backgammon Program, Achieves Master-Level Play. Applications of Neural Networks **6**(2), 215–219 (1994). DOI 10.1162/neco.1994.6.2.215. URL https://doi.org/10.1162/neco.1994.6.2.215
30. Todorov, E., Li, W.: A generalized iterative LQG method for locally-optimal feedback control of constrained nonlinear stochastic systems. In: American Control Conference, pp. 300–306 vol. 1 (2005). DOI 10.1109/ACC.2005.1469949
31. Wang, Z., Schaul, T., Hessel, M., Van Hasselt, H., Lanctot, M., De Freitas, N.: Dueling Network Architectures for Deep Reinforcement Learning. In: Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48, ICML'16, pp. 1995–2003. JMLR.org (2016). URL http://dl.acm.org/citation.cfm?id=3045390.3045601
32. Watkins, C.J.C.H.: Learning from delayed rewards. Ph.D. thesis, King's College, Cambridge (1989)
33. Williams, R.J.: Simple statistical gradient-following algorithms for connectionist reinforcement learning. Machine Learning **8**(3), 229–256 (1992). DOI 10.1007/BF00992696. URL https://doi.org/10.1007/BF00992696
34. Wu, Y., Mansimov, E., Grosse, R.B., Liao, S., Ba, J.: Scalable trust-region method for deep reinforcement learning using Kronecker-factored approximation. In: I. Guyon, U.V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, R. Garnett (eds.) Advances in Neural Information Processing Systems 30, pp. 5279–5288. Curran Associates, Inc. (2017). URL http://papers.nips.cc/paper/7112-scalable-trust-region-method-for-deep-reinforcement-learning-using-kronecker-factored-approximation.pdf