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

Model-free Deep Reinforcement Learning – Methods and Applications

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 to MDPs.

2 Background

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:

$$G_t = \mathbb{E}\left[\sum_{\tau=t}^{\infty} \gamma^{\tau-t} \mathcal{R}(s_\tau, a_\tau) \middle| s_\tau, a_\tau\right]$$

Here $\gamma \in [0, 1)$ describes the discount factor, which trades of immediate and future rewards.
RL algorithms now aim to find a policy $\pi : \mathcal{S} \to \mathcal{A}$, which describes the optimal

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

behavior to maximize the expected future reward from all states. Consequently, an action-value function, which returns the value of taking an action a in state s, under the policy π is defined as:

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

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 $V^*(s) = \mathbb{E}\left[G_t | s_t = s, \pi\right] = \max_a Q^*(s,a)$ can be used, e.g. in combination with a model of the environment.
In contrast to value-based methods, policy-based methods directly parameterize a policy $\pi(s;\theta)$. One such method is REINFORCE [44], it updates the policy parameters θ in the direction $\nabla_\theta \mathbb{E}[G_t] \approx \nabla_\theta \log \pi(s_t;\theta) G_t$. In order to reduce variance and keeping the estimate unbiased, a baseline $b(s_t)$ is subtracted from the return $\nabla_\theta \log \pi(s;\theta)(G_t - b(s_t))$. For this baseline, it is common to use an estimate of the value function $b(s_t) \approx V_\pi(s_t)$. By applying the policy gradient theorem, $G_t - b(s_t)$ can be seen as an estimate of the advantage function $A_\pi(s_t,a_t) = Q_\pi(s_t,a_t) - V_\pi(s_t)$. Combining parameterized policies and value function estimator results in *actor-critic* (AC) methods.

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 the current deep RL landscape including improvements and 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. [38, chapter 5]

4.1 Off-Policy

One of the earlier off-policy algorithms, named *neural fitted Q-iteration* (NFQ), was published by Riedmiller [27] in 2005. NFQ works model-free based on vanilla Q-Learning [43] and approximates the action-value function with a multilayer-perceptron. Compared to other approaches at the time, NFQ allows to learn with

a relatively good data efficiency. Further, batches from a replay memory [19] are used in order to train the multilayer-perceptron with *resilient backpropagation* (RPROP) [28]. 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 [39] the arguably biggest improvement in NN based RL – *Deep Q-Networks* (DQN) – were introduced by Mnih, et al. [24]. They built upon their prior results [23] on the Atari Arcade Environment [5] 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* (CNN). Benchmarks of the more recent DQN version [24] show an improved performance as well as a better generalization to more Atari games compared to initial work [23]. 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. [11] show that DQNs [24] are 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, DDQN utilizes the already existing online Q-network for determining the greedy policy and the target Q-network for estimating its value. Further, this allows to achieve better performance on most of the Atari games. Decoupling is also done in a more extreme form with *Dueling Double DQNs* (DDDQN) [42]. They introduce two separate function approximations, one for the state-value function and one for the state-dependent action-advantage function, in order to represent the action-value[1]. These estimators allow the dueling architecture to learn about the value of a state regardless of the action taken. This is especially important when the action has no repercussions about the near future. Combining DDQN with gradient clipping and *prioritized experience replay* (PER) [29] shows that sampling rare experiences with a higher probability makes learning with replay memories more efficient [19]. 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 occurs 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 *hierarchical-DQN* (h-DQN) [17], 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 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.

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

As the above results show, Q-learning variations are 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) [8] approach both problems with a DQN based algorithm. Similar to DDDQN [42], 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) [40] 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 [37]. 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. [18] also adapt the underlying idea of DQN [24] 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 AC algorithm with approximate Q-Learning based on deterministic policy gradients (DPG) [35]. DQN's [24] architecture is improved by adding batch normalization layers [14] to deal with different physical units in the observation space. In order to stabilize policies, they adapt the idea of target networks from DQN [24] and use "soft" target updates for AC. Albeit, this stability is, according to Haaranoja, et al. [9], extremely difficult to achieve and further, DDPG is brittle to hyperparameter choices. As consequence, Haaranoja, et al. present soft actor critic (SAC) [9]. They combine the AC approach from DDPG [18] with maximum entropy reinforcement learning, which enables exploration and stability. Similarly to NAF [8], 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. *Twin delayed DDPG* (TD3) [6] was developed in parallel to SAC and continues to addresses the hyperparameter sensitivity, but further shows that overestimation of the action-value function is also a pressing issue for AC methods. However, for the continuous actions space the approach of DDQN [11] is not easily applicable. Therefore, TD3 proposes three key improvements. A clipped variant of double Q-Learning [10] trains two separate Q-Networks in order to reduce the overestimation bias. Policy smoothing, similar to expected SARSA [32], adds noise to the target action, which avoids 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 reduces the risk of optimizing the policy based on incorrect value estimates and hence the policy 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 DDPG with distributed actors, such as the on-policy *asynchronous advantage actor critic* (A3C) [22], and a distributional value estimate [4] results in *Distributional Distributed DDPG* (D4PG) [3]. A3C also has an off-policy counterpart *actor-critic with experience replay* (ACER) [41], which outperforms A3C by significant margin regarding performance and sam-

ple efficiency. This is achieved by Retrace [25] Q-value estimation, which reduces the bias of policy gradient estimates, whereas truncated importance weights help to reduce variance. Additionally, maintaining a running average of past policies forces the new policy to stay close to this average. This is similar to computing the *Kullback-Leibler* (KL) *trust region policy optimization* (TRPO), however it is computationally more efficient. The above algorithms used multiple agents to provide better exploration and faster convergence, however the goal was to find the optimal policy for one worker. *Multi-agent DDPG* (MADDPG) [20] on the other hand is interested in using multiple agents to collaborate and/or compete. The decentralized actors are only trained with local information, which allows for easier inference, the centralized critics have access to all information during training. In order to reduce variance due to interactions with other actors, policy ensembles are proposed.

4.2 On-Policy

Comparably important to DQN [24] in the off-policy setting is the introduction of *asynchronous advantage actor critic* (A3C) [22] for the on-policy setting. Besides A3C, Mnih, et al. also introduce asynchronous off-policy methods for Q-Learning and SARSA. However, they show that A3C outperforms the proposed off-policy methods. Multiple actor threads allow A3C to collect more experience in a larger space of the environment and incorporate it into one shared policy 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 [24].[2] A3C was also combined with GPU computation [2]. Similar to SAC [9] entropy is included in objective function, however A3C utilizes it for regularization and not as constraint. Aside from the discrete Atari games, A3C was also tested for continuous action spaces and was able to learn reliable policies. Following the concept of policy gradients, Schulman, et al. introduce trust region policy optimization (TRPO) [30]. TRPO can be seen as combination of minorization-maximization [13] and *natural actor critic* (NAC) [26] 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 optimizes its policy by enforcing a trust region constraint. The 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 be more computationally efficient, the natural policy gradient [16] is computed approximately with the *Conjugate Gradient* (CG) algorithm. Further, due to the multiple approximations, before applying the update, a backtracking line search determines whether the update still satisfies the KL constraint or not. One big drawback of TRPO in practice is its sample efficiency. Theoretically, TRPO can be applied to any policy, but it is often not practical to use deep neural networks policies. This shortcoming is addressed by Proximal Policy Optimization (PPO) [31]. PPO keeps all monotonic improvement guarantees while using first order optimization by replacing the KL

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

constraint with KL regularization. Addtionally, the importance sampling weights are clipped, which results in a lower bound for the unclipped objective. PPO can also be combined with A3C's idea of distributed actors resulting into *Distributed PPO* (DPPO) [12]. Besides PPO, *actor-critic using kronecker-factored trust region* (ACKTR) [45] also approaches TRPO's problem of sample complexity. ACKTR utilizes A3C's advantage function and approximates the natural gradient with *kronecker-factored approximate curvature* (K-FAC) [7, 21], which offers a comparable cost to SGD. By maintaining a running average of curvature information, K-FAC is able to reduce variance and achieve scalability, therefore ACKTR is more efficient in computing the inverse Fisher Information matrix. 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 [22].

5 Applications in Games and Robotics

One of the earliest success stories of Deep RL was TD-gammon [39] achieving master level performance in backgammon. TD-gammon was starting with zero prior knowledge and improved by solely playing against itself. However, consecutive experiments aiming to reproduce the success in backgammon for other games such as chess, Go, checkers failed. This was often attributed to the idea that state space exploration was helped by the stochasticity of the dice rolls [**?**]. Albeit these failures, DQN was motivated by this success a showed that it was not only able to outperform existing methods on most of the Atari games, but even surpass human-level performance. RL algorithms following after DQN were able to improve the performance even further for all games (e.g. A3C) or specifically for poorly performing games, such as h-DQN on Montezuma's Revenge. Based on the idea of TD-gammon, AlphaGo [33] combined self-play with CNNs, supervised learning and *Monte Carlo Tree Search* (MCTS). In its core AlphaGo is selecting moves by a novel MCTS, which is guided by learned a value function and policy. However, instead of directly training from scratch, value function and policy are pretrained on human expert moves and are afterwards improved by self-play. In 2016 AlphaGo showed significantly better performance compared to other programs and was able to achieve the first victories over professional players. AlphaGo Zero [36] eliminated the supervised learning aspect from AlphaGo and used no human data or guidance beyond the basic rules of the game. Additional improvements include an improved version of MCTS and the use of MCTS during training/self-play. AlphaGo Zero was able to defeat the strongest human players without a single loss and also beat AlphaGo considerably. AlphaZero [34] generalizes AlphaGo Zero to the chess and shogi domain. As a consequence, AlphaZero has to take different outcomes, e.g. draws, into account, further it is not able to exploit symmetries such as in Go. One major criticism is the underlying computational power, e.g. training for AlphaZero was executed on 5,000 TPUs [15]. Therefore, it is questionable if the same algorithm performs equally well with less computational power.

6 problems

convergence hyperparams forgetting ???

References

1. Abdolmaleki, A., Springenberg, J.T., Tassa, Y., et al.: Maximum a Posteriori Policy Optimisation (2018). URL http://arxiv.org/abs/1806.06920
2. Babaeizadeh, M., Frosio, I., Tyree, S., et al.: Reinforcement Learning through Asynchronous Advantage Actor-Critic on a GPU (2017). URL http://arxiv.org/abs/1611.06256
3. Barth-Maron, G., Hoffman, M.W., Budden, D., et al.: Distributed Distributional Deterministic Policy Gradients. In: International Conference on Learning Representations (2018). URL http://arxiv.org/abs/1804.08617
4. Bellemare, M.G., Dabney, W., Munos, R.: A Distributional Perspective on Reinforcement Learning. In: International Conference for Machine Learning (ICML) (2017). URL http://arxiv.org/abs/1707.06887
5. Bellemare, M.G., Naddaf, Y., Veness, J., et al.: 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
6. 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
7. 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
8. Gu, S., Lillicrap, T., Sutskever, I., et al.: 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
9. Haarnoja, T., Zhou, A., Hartikainen, K., et al.: Soft Actor-Critic Algorithms and Applications (2018). URL http://arxiv.org/abs/1812.05905
10. van Hasselt, H.: Double Q-learning. In: 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
11. 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
12. Heess, N., TB, D., Sriram, S., et al.: Emergence of Locomotion Behaviours in Rich Environments (2017). URL http://arxiv.org/abs/1707.02286
13. Hunter, D.R., Lange, K.: A tutorial on MM algorithms. The American Statistician **58**(1), 30–37 (2004)
14. 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
15. Jouppi, N.P., Young, C., Patil, N., et al.: In-Datacenter Performance Analysis of a Tensor Processing Unit. In: International Symposium on Computer Architecture (ISCA), ISCA '17, pp. 1–12. ACM, New York, NY, USA (2017). DOI 10.1145/3079856.3080246. URL http://doi.acm.org/10.1145/3079856.3080246
16. 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
17. Kulkarni, T.D., Narasimhan, K., Saeedi, A., et al.: Hierarchical Deep Reinforcement Learning: Integrating Temporal Abstraction and Intrinsic Motivation. In: 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
18. Lillicrap, T.P., Hunt, J.J., Pritzel, A., et al.: Continuous control with deep reinforcement learning. In: International Conference on Learning Representations (ICLR) 2016. London, UK (2016). URL http://arxiv.org/abs/1509.02971
19. 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

20. Lowe, R., WU, Y.I., Tamar, A., et al.: Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. In: Conference on Neural Information Processing Systems (NIPS), pp. 6379–6390. Curran Associates, Inc., Long Beach, CA, USA (2017). URL http://papers.nips.cc/paper/7217-multi-agent-actor-critic-for-mixed-cooperative-competitive-environments.pdf
21. Martens, J., Grosse, R.: Optimizing Neural Networks with Kronecker-factored Approximate Curvature (2015). URL http://arxiv.org/abs/1503.05671
22. Mnih, V., Badia, A.P.A.P., Mirza, M., et al.: Asynchronous Methods for Deep Reinforcement Learning. In: 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
23. Mnih, V., Kavukcuoglu, K., Silver, D., et al.: Playing Atari with Deep Reinforcement Learning. NIPS Deep Learning Workshop 2013 (2013). URL http://arxiv.org/abs/1312.5602
24. Mnih, V., Kavukcuoglu, K., Silver, D., et al.: Human-level control through deep reinforcement learning. Nature **518**(7540), 529–533 (2015). DOI 10.1038/nature14236. URL http://www.nature.com/articles/nature14236
25. Munos, R., Stepleton, T., Harutyunyan, A., et al.: Safe and Efficient Off-Policy Reinforcement Learning. In: Advances in Neural Information Processing Systems 29, pp. 1054–1062. Curran Associates, Inc. (2016). URL http://papers.nips.cc/paper/6538-safe-and-efficient-off-policy-reinforcement-learning.pdf
26. Peters, J., Schaal, S.: Natural actor-critic. Neurocomputing **71**(7-9), 1180–1190 (2008)
27. Riedmiller, M.: Neural Fitted Q Iteration - First Experiences with a Data Efficient Neural Reinforcement Learning Method. In: Machine Learning: ECML 2005, pp. 317–328. Springer Berlin Heidelberg, Berlin, Heidelberg (2005)
28. 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)
29. Schaul, T., Quan, J., Antonoglou, I., et al.: Prioritized Experience Replay. In: Interantional Conference for Learning Representations (ICLR) (2015). URL http://arxiv.org/abs/1511.05952
30. Schulman, J., Levine, S., Abbeel, P., et al.: Trust Region Policy Optimization. In: 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
31. Schulman, J., Wolski, F., Dhariwal, P., et al.: Proximal Policy Optimization Algorithms (2017). URL http://arxiv.org/abs/1707.06347
32. van Seijen, H., van Hasselt, H., Whiteson, S., et al.: 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
33. Silver, D., Huang, A., Maddison, C.J., et al.: Mastering the game of Go with deep neural networks and tree search. Nature **529**(7587), 484–489 (2016). DOI 10.1038/nature16961. URL http://www.nature.com/articles/nature16961
34. Silver, D., Hubert, T., Schrittwieser, J., et al.: Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm pp. 1–19 (2017). DOI 10.1002/acn3.501. URL http://arxiv.org/abs/1712.01815
35. Silver, D., Lever, G., Heess, N., et al.: 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
36. Silver, D., Schrittwieser, J., Simonyan, K., et al.: Mastering the game of Go without human knowledge. Nature **550**(7676), 354–359 (2017). DOI 10.1038/nature24270. URL http://www.nature.com/doifinder/10.1038/nature24270
37. 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)
38. 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
39. 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

40. 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
41. Wang, Z., Bapst, V., Heess, N., et al.: Sample Efficient Actor-Critic with Experience Replay. In: International Conference on Learning Representations (ICLR) (2017). URL http://arxiv.org/abs/1611.01224
42. Wang, Z., Schaul, T., Hessel, M., et al.: 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
43. Watkins, C.J.C.H.: Learning from delayed rewards. Ph.D. thesis, King's College, Cambridge (1989)
44. 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
45. Wu, Y., Mansimov, E., Grosse, R.B., et al.: Scalable trust-region method for deep reinforcement learning using Kronecker-factored approximation. In: 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