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

Policy Gradient Estimation

A survey on policy gradient estimators

Mattis Manfred Kämmerer · Mahdi Enan · Simon Kiefhaber

Received: date / Accepted: date

Abstract Policy gradient methods use a policy model to decide which actions maximize the expected accumulated reward received in a given environment. Policies can be deterministic but are often stochastic to accommodate for the nature of the given environment. One class of problems that is frequently chosen to evaluate these methods are continuous control problems. Depending on the environment the formal design of an optimal controller can be infeasible or very labor-intensive. Policy gradient methods can provide useful control models while aiming to optimize their data efficiency, thus requiring less resources for the learning process. The efficiency of these methods highly depends on the direction of change, i.e., the gradient they estimate. The goal of policy gradient estimation is to estimate a function that predicts how the policy should be changed to solve the reinforcement learning problem of maximizing the accumulated reward of an agent. Thus, different functions for estimating the gradient have been proposed, some of which outperform others in specific areas. This survey gives a general introduction to policy gradient, and gives an overview of existing gradient estimation approaches.

Keywords reinforcement learning · policy gradients · actor-critic · advantage estimation · continuous control

1 Introduction

Introductions to reinforcement learning often start by introducing value-based methods. They aim to gain the maximum accumulated reward in a Markov Decision Process (MDP). Value-based methods estimate the expected value of each action in each state, approximating a function that ranks actions in a given state, which is then used to derive a policy. Though this works well for problems with countable actions and states, its performance is not stable on continuous control problems. Even with optimal hyperparameters, the outcomes vary between separate trainings. [12] In constract to value-based methods, the other main class of model-free reinforcement learning algorithms is called policy-based methods, or

Mattis Manfred Kämmerer · Mahdi Enan · Simon Kiefhaber
TU Darmstadt

policy gradient methods. Research has shown, that policy gradient methods can reach better performance in these environments. [16] Using a parameterized (often stochastic) policy to decide the next action, they can easily incorporate prior domain knowledge, and reuse some of the value-based approaches for improved performance, but require a lot of configuration to produce an effective agent for a specific environment. Also, they frequently require on-policy training and often converge slower in discrete settings. Options to model a policy gradient method are not only given by the policy model itself, but also in the way a policy improvement is computed. [17] In this case, improvement means a step in the parameter space such that the policy under the new parameters will on average perform better than the old policy. Policy gradient estimation is the term we use to describe the process of computing the next step in parameter space. The goal is to estimate a function that predicts how the policy should be changed to maximize the accumulated reward of an agent in an MDP. Since this is the core problem of policy gradient methods, it is also the main topic of this paper.

In section 2, we give some preliminaries and describe the problem setup in detail. In section 3, we discuss different approaches to estimate the policy gradient. Using our insights from section 3, we derive the actor-critic framework in section 4, which harnesses some value-based approaches for improved gradient updates. Then, in section 5, we introduce some gradient ascent methods that build on the approaches given in 3, refining the gradient estimation by Fisher's information matrix to get the natural gradient. Finally, in section 6, we give a conclusion of the reviewed approaches, and discuss potential areas of improvement.

2 Preliminaries

Definitions. We define states $s \in \mathbb{X}$, actions $a \in \mathbb{U}$, and rewards $r \in \mathbb{R}$. A trajectory $\rho^h_{\pi_\theta} := (s_0, a_0, s_1, a_1, \ldots, s_h, a_h)$ is generated by drawing $s_0 \sim p(x_0)$ according to the distribution over initial states $p(x_0)$, and successively sampling $a_t \sim \pi_\theta(a_t|s_t)$ according to the policy π_θ, and $s_{t+1} \sim p(s_{t+1}|s_t, a_t)$ until the horizon h is reached. At each time step, we receive a reward according to $r_t = r(s_t, a_t, s_{t+1})$. A trajectory can also be called *roll-out* or *episode*, though the term episode implies it ends in a terminal state.

We assume a Markov Decision Process (MDP), meaning the proability distribution of the next states is independent of past states $s_{0:t-1}$ given the present state s_t and action a_t,

$$p(s_{t+1}|s_t, a_t) = p(s_{t+1}|s_{0:t}, a_{0:t}). \tag{1}$$

Where we define $i : j$ with $i, j \in \mathbb{N}, i < j$ as an index over all integers from i to j, i.e., $s_{i:j} := s_i, s_{i+1}, \ldots, s_j$.

Distributions often used as policies are Gibbs policies $\pi_\theta(a|s) = \frac{exp(\phi(s,a)^T\theta)}{\sum_b exp(\phi(s,b)^T\theta)}$ [19,5] for discrete problems, and Gaussian policies $\pi_\theta(a|s) = N(\phi(s,a)^T\phi_1, \phi_2)$ for continuous problems, where ϕ_2 is an exploration parameter [21,20]. $\phi(s,a)$ is the vector of basis functions on the state-action pair.

Policy gradient. Our goal with respect to episodes is to maximize the expectation of the total reward $\mathbb{E}_{s\sim p(s|\theta)}[\mathcal{R}^{\infty}]$. The total reward in the horizon h for the non-discounted problem is $\sum_{t=0}^{h} r_t$. For most cases, a discount factor $\gamma \in [0,1)$ is added to ensure that this sum is finite for the infinite horizon $h \to \infty$. Intuitively, this reflects the idea that the relevance of later actions declines, and reduces variance during learning. The discounted total reward is

$$\mathcal{R}^h := \sum_{t=0}^{h} \gamma^t r_t. \tag{2}$$

Since we have only empirical knowledge of the performance of the policy, we need to approximate an optimal policy by estimating a gradient. Thus, we search $\nabla_\theta J(\theta) := \nabla_\theta \mathbb{E}_{s\sim p(s|\theta)}[\psi(s,a)]$, to make a policy gradient step according to $\theta_{k+1} = \theta_k + \alpha_k \nabla_\theta J(\theta)$, where ψ denotes some score function, see section 3, and α_k denotes a learning rate. It is shown that for $\lim_{k\to\infty} \alpha_k = 0$, and $\sum_{k=0}^{\infty} \alpha_k$ we are guaranteed to converge to a local optimum [19].

Likelihood-ratio gradients. Using the definition of the expectation, we derive

$$\begin{aligned}
\nabla_\theta \mathbb{E}_{s\sim p(s|\theta)}[\psi(s,a)] &= \nabla_\theta \int_{\mathbb{X}} p(s|\theta) \int_{\mathbb{U}} \pi_\theta(a|s)\psi(s,a) da ds \\
&= \int_{\mathbb{X}} p(s|\theta) \int_{\mathbb{U}} \nabla_\theta \pi_\theta(a|s)\psi(s,a) da ds \\
&= \int_{\mathbb{X}} p(s|\theta) \int_{\mathbb{U}} \pi_\theta(a|s)\nabla_\theta \log \pi_\theta(a|s)\psi(s,a) da ds.
\end{aligned}$$

Where we used $\nabla_\theta \pi_\theta(a|s) = \pi_\theta(a|s)\nabla_\theta \log \pi_\theta(a|s)$, obtained from the likelihood ratio, which gives us the likelihood-ratio gradient,

$$\nabla_\theta \mathbb{E}_{\pi_\theta}[\psi(s,a)] = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a|s)\psi(s,a)]. \tag{3}$$

intuitively meaning we should to increase the probability of actions judged as good by the score function ψ. We present some frequently used score functions in section 3.

3 Policy Gradient Estimation

In this section, we discuss some approaches to calculate the estimated policy gradient $\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a|s)\psi(a|s)]$ using different score functions to substitute ψ.

Immediate updates. The first class of algorithms developed to update a policy directly instead of deriving it from a value function approximation were REINFORCE [21] algorithms. REINFORCE uses immediate rewards with a reinforcement baseline b in its gradient steps:

$$\psi_t \leftarrow r_t - b. \tag{4}$$

Obviously, this does not involve temporal information. To credit temporal information ψ_t needs to involve multiple time steps, meaning we need to delay updates until we have enough information to evaluate the gradient.

Episodic updates. A common way of crediting temporal information in score function is to sample trajectories, also called roll-outs, and to updates using the complete roll-out. We call these approaches episodic. When we do this, we sum up the immediate updates, which we can rearrange to get

$$\sum_{t=0}^{h} \nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{h} \nabla_\theta \log \pi_\theta(a|s)\psi(s,a)\right]. \tag{5}$$

Episodic REINFORCE [21], the first approach to this concept performs roll-outs and takes only the final accumulated reward:

$$\sum_{t=0}^{h} \nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{h} \nabla_\theta \log \pi_\theta(a|s)\mathcal{R}^h\right]. \tag{6}$$

However, taking this a step further, it should be possible to attribute the reward at each step to the respective action. A very basic episodic score function that fits into our framework in (5) is to use the discounted rewards, similar to (2):

$$\psi_t \leftarrow \gamma^t r_t. \tag{7}$$

If we only take batches of episodes starting at time t', we can also use only the rewards following a chosen action $a_{t'}$ by calculating only $\sum_{t=t'}^{h} \nabla_\theta J(\theta)$ with $\psi_t \leftarrow \gamma^{t'} r_t$. As we will see, this concept is also used in value functions (9), and (10).

Given $r_t > 0, \forall t = 0, \ldots, h$, these approaches can only increase action probabilities. Obviously, we normalize to ensure $\forall s \in \mathbb{X} : \int_{\mathbb{U}} \pi(a|s)da = 1$. This means that actions can only become less probable in relation to other actions. We find that this introduces a lot of variance when learning from samples [19]. One approach to counter the variance is to introduce a baseline

$$\psi_t \leftarrow \gamma^t (r_t - b(s_t)) \tag{8}$$

The baseline can represent any prior knowledge or assumption about states, e.g., their mean value. Though we find that there is a specific way to use this baseline which fits surprisingly well into this framework.

Value functions. Given we know the actual value, i.e. the expected accumulated reward we will get when we start in a state s_t, this function can be used to evaluate the performance of our policy.

$$V^\pi(s_t) := \mathbb{E}_{\substack{s_{t+1:h} \\ a_{t:h}}}\left[\sum_{l=0}^{h} \gamma^{t+l} r_{t+l}\right]. \tag{9}$$

Though, in general we need to approximate this function. This is the core idea of value-function approximation, but is also important in the context of policy gradients, especially for actor-critic approaches, which we discuss in 4.

Another frequently used function is the state-action value function. Instead of the expected accumulated reward starting from state s_t, this function gives the expected accumulated reward given an action a_t is selected

$$Q^\pi(s_t, a_t) := \mathbb{E}_{\substack{s_{t+1:h} \\ a_{t+1:h}}}\left[\sum_{l=0}^{h} \gamma^{t+l} r_{t+l}\right]. \tag{10}$$

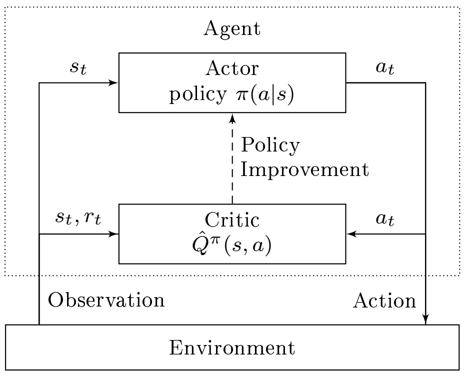

Fig. 1 An actor-critic framework comparable to Kimura et al. [9].

Approximating this function by an unbiased estimator $f_w(s_t, a_t) = \hat{Q}^\pi(s_t, a_t)$, we can also use this estimation as a score function $\psi \leftarrow \hat{Q}$. Sutton et al. [19] show that using this function approximation we will converge to the same local optimum. However, they also find this requires $\int_{\mathbb{X}} \pi(s,a) f_w(s,a) da = 0$ for each state. Given this assumption, the function estimator f_w resembles an advantage function

$$A^\pi(s_t, a_t) := Q^\pi(s_t, a_t) - V^\pi(s_t) \tag{11}$$

as in [1].

Whenever we require estimating a value function for updating our policy, we can name the policy actor, and the estimated value function critic. From this observation, we can define a class of policy optimization methods called actor-critic methods, see section 4.

4 Actor-Critic Methods

Policy gradient methods can be described in terms of two main steps often called policy evaluation and policy improvement. For actor-critic approaches, we separate these steps from the actor component by implementing a critic. This means, the actor consists only of the policy and how to improve it given feedback from the critic, which is focused on estimating a score for the actions taken. By that concept, observations of s_t from the environment are given to the actor only to decide the next action, and to the critic only to improve its function estimation. Figure 1 shows the general structure of an actor-critic algorithm.

Critic. The critic estimates a value function as defined in (9). [19], and [11] find that the estimation $f_w^\pi(s,a) \approx Q^\pi(s,a)$ does not affect the unbiasedness of the gradient estimate under some restrictions. Specifically, this holds for $f_w^\pi(s,a) = \nabla_\theta \log \pi(a|s)^T w$, thus $f_w^\pi(s,a)$ being a linear function parameterized by the vector w. However, as we have seen in the last section, 3, this function estimator will have a mean of zero $\int_{\mathbb{U}} \nabla_\theta \pi(a|s) du$, thus it estimates an advantage function 11. However, to estimate the advantage function, we require $V^\pi(s)$. We will show how this leads to a new problem but still, this insight guarantees that function approximation does not cause divergence, and really enables recent research in reinforcement learning for continuous control problems, e.g., in humanoid robotics.

Actor. The main challenge we tackle in the actor other than choosing an actual policy is how to do the policy improvement step. Though, one could argue that this step is somewhere in between the critic and the actor.

Traditionally, the improvement is often done by TD [18], i.e., we use the temporal difference between the critic's estimations

$$r_t + \gamma \hat{V}^\pi(s_{t+1}) - \hat{V}^\pi(s_t). \tag{12}$$

This is where we need to come back to the insight that from $f_w^\pi(s,a) = \nabla_\theta \log \pi(a|s)^T w$, it follows that $f_w^\pi(s,a)$ estimates the advantage function, and we require some form of bootstrapping for $V^\pi(s,a)$. If we use temporal difference in this context, we run into a bootstrapping problem, as (11) subtracts $\hat{V}^\pi(s_t)$, meaning we would only learn immediate rewards [15]. This would render the process biased. Sutton et al. [19] and Konda et al. [10] suggest estimating an action value function as in (10). We can approximate this f_w^π by least-squares optimization over multiple $\hat{Q}^\pi(s,a)$ obtained from roll-outs. However, Peters et al. [14] find that this approximation is highly reliant on the distribution of the training data. This comes from the realization, that we use only a subspace of the actual action value function in f_w^π, which is only a state value function. One can compare this to approximating a parabola by a line, whereby the approximation changes wildly depending on which part of the parabola is in the training data. An approach to solve this bootstrapping problem is to rewrite the Bellman Equation using (11) and (10) with $A^\pi(s,a) = f_w^\pi(s,a)$, $V^\pi(s) = \phi(s)^T v$, and a zero-mean error term $\epsilon(s_t, a_t, s_{t+1})$, we get

$$Q^\pi(s,a) = A^\pi(s,a) + V^\pi(s) = r(s,a) + \gamma \int_{\mathbb{X}} p(s'|s,a) V^\pi(s') ds' \tag{13}$$

$$\nabla_\theta \log \pi(a_t|s_t)^T w + \phi(s_t)^T v = r(s_t, a_t) + \gamma \phi(s_{t+1})^T v + \epsilon(s_t, a_t, s_{t+1}) \tag{14}$$

which gives involves only linear equations to solve. [14]

With these insights in mind, 5 presents the natural gradient, a refined type of gradient which has a convenient fit in the actor-critic setting we just established.

5 Natural Gradient

Natural gradients were at first proposed for use in supervised learning settings by Amari et al. [4], but have been shown to be effective in reinforcement learning by Kakade [8] and Peters et al. [14].

When using normal gradient steps, we find that steps can become very small when a plateau is reached. This can will drastically slow down the learning process, and in the worst case cause some algorithms to prematurely. However, we can use some additional information to refine the gradient. Figure 2 shows an example by Peters et al. [15] that gives a visual intuition about the difference between 'vanilla' and natural policy gradients.

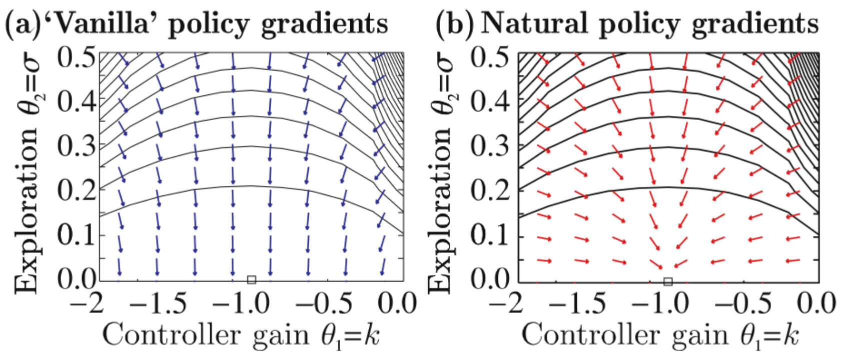

Fig. 2 An experiment showing where the natural gradient has a great advantage. [15]

Using the Fisher information matrix F_θ, and the gradient estimate we discussed in section 3 gives us the definition

$$\widetilde{\nabla}_\theta J(\theta) := F_\theta^{-1} \nabla_\theta J(\theta) \tag{15}$$

of the natural gradient. The Fisher information matrix represents the certainty we have on our estimate of the gradient and is defined as the covariance of the log likelihood function of a trajectory $\rho^h_{\pi_\theta}$, which as [14] show can be written as

$$F_\theta = \int_{\mathbb{X}} p(s|\theta) \int_{\mathbb{U}} \pi_\theta(a|s) \nabla_\theta \log \pi_\theta(a|s) \nabla_\theta \log \pi_\theta(a|s)^T dads. \tag{16}$$

When we recall the definition (3) of likelihood-ratio gradients, we see that setting ψ as $\nabla_\theta \log \pi(a|s)^T w$, we get

$$\nabla_\theta J(\theta) = \int_{\mathbb{X}} p(s|\theta) \int_{\mathbb{U}} \pi_\theta(a|s) \nabla_\theta \log \pi(a|s) \nabla_\theta \log \pi(a|s)^T dads\ w := F_\theta w. \tag{17}$$

From (15), (16), and (17), it follows that

$$\widetilde{\nabla}_\theta J(\theta) = F_\theta^{-1} \nabla_\theta J(\theta) = F_\theta^{-1} F_\theta w = w. \tag{18}$$

Thus, this approach does not require an actual estimate of the Fisher information matrix, but only an estimate of w, with the update step according to $\theta_{k+1} = \theta_k + \alpha_k w$.

Peters et al. [14] present this idea and suggest LSTD(λ)-Q, a version of least-squares temporal difference learning [6], as well as episodic natural actor-critic.

Note that this approach abuses some specific conveniences about the problem setup to find an elegant solution. However, this is not the only way to approach this problem. In this paper, we focus on one presentation to present the core idea, but the field of policy gradient optimization is surprisingly resourceful. In the conclusion 6, we attempt to show the versatility and relevance in current research, and give some directions for further reading.

6 Conclusion

In this paper, we have introduced policy gradient methods as a class of reinforcement learning algorithms. We show why policy gradient methods are effective in these environments, and we give some intuitions about the concept. Further, we show the core elements of policy gradient methods, and we discuss some intricacies that they bring, and some research development in the attempts of improving the efficiency and stability of policy gradients.

There are many possibilities for improvement as it can be seen as a very modularized and flexible. We find that the theoretical basis for policy gradient focused approaches reaches far back with some fundamental algorithms like Levenberg-Marquardt [13], and the first presentation of REINFORCE [21]. Policy gradient methods were shown to be increasingly potent by authors such as Sutton [19], and still provide space for a lot of new work being published. Algorithms such as the aforementioned Natural Actor-Critic [14] have been showing some of their potentials, but even so potential areas of improvement. Schulman et al. [17] present a form of advantage estimation using γ as a parameter for bias-variance trade-off. Fellows et al. [7] use Fourier analysis to analytically improve policy gradients. Abdolmaleki et al. [3] introduce MPO, an off-policy algorithm using expectation maximization, related to the approach of TRPO [16]. Yet a different approach is given by Zhang et al. [22], who formulate policy optimization as a Wasserstein gradient flow problem, which moves into a Wasserstein manifold in the distribution space. This is comparable to approaches by Abdolmaleki et al. [2], which introduces a bound on the Kullback-Leibler divergence during policy search. From the developments in recent research, it is fair to say that policy gradient methods play a major role in reinforcement learning.

References

1. Advantage updating. Tech. Rep. WL–TR-93-1146, Wright-Patterson Air Force Base Ohio: Wright Laboratory (1993). URL http://leemon.com/papers/1993b.pdf
2. Abdolmaleki, A., Lioutikov, R., Peters, J.R., Lau, N., Pualo Reis, L., Neumann, G.: Model-based relative entropy stochastic search. In: C. Cortes, N.D. Lawrence, D.D. Lee, M. Sugiyama, R. Garnett (eds.) Advances in Neural Information Processing Systems 28, pp. 3537–3545. Curran Associates, Inc. (2015). URL http://papers.nips.cc/paper/5672-model-based-relative-entropy-stochastic-search.pdf
3. Abdolmaleki, A., Springenberg, J.T., Tassa, Y., Munos, R., Heess, N., Riedmiller, M.: Maximum a posteriori policy optimisation. In: International Conference on Learning Representations (2018). URL https://openreview.net/forum?id=S1ANxQW0b
4. Amari, S.I.: Natural gradient works efficiently in learning. Neural Comput. **10**(2), 251–276 (1998). DOI 10.1162/089976698300017746. URL http://dx.doi.org/10.1162/089976698300017746
5. Bagnell, J.A.: Learning decisions : Robustness , uncertainty , and appoximation (2004)
6. Boyan, J.A.: Least-squares temporal difference learning. In: Proceedings of the Sixteenth International Conference on Machine Learning, ICML '99, pp. 49–56. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1999). URL http://dl.acm.org/citation.cfm?id=645528.657618
7. Fellows, M., Ciosek, K., Whiteson, S.: Fourier policy gradients. In: ICML 2018: Proceedings of the Thirty-Fifth International Conference on Machine Learning (2018). URL http://www.cs.ox.ac.uk/people/shimon.whiteson/pubs/fellowsicml18.pdf
8. Kakade, S.: A natural policy gradient. In: T.G. Dietterich, S. Becker, Z. Ghahramani (eds.) Advances in Neural Information Processing Systems 14 (NIPS 2001), pp. 1531–1538. MIT Press (2001). URL http://books.nips.cc/papers/files/nips14/CN11.pdf

9. Kimura, H., Kobayashi, S.: An analysis of actor/critic algorithms using eligibility traces: Reinforcement learning with imperfect value function. In: ICML (1998)
10. Konda, V.R., Tsitsiklis, J.N.: Actor-critic algorithms. In: S.A. Solla, T.K. Leen, K. Müller (eds.) Advances in Neural Information Processing Systems 12, pp. 1008–1014. MIT Press (2000). URL `http://papers.nips.cc/paper/1786-actor-critic-algorithms.pdf`
11. Konda, V.R., Tsitsiklis, J.N.: On actor-critic algorithms. SIAM J. Control Optim. **42**(4), 1143–1166 (2003). DOI 10.1137/S0363012901385691. URL `https://doi.org/10.1137/S0363012901385691`
12. Metz, L., Ibarz, J., Jaitly, N., Davidson, J.: Discrete sequential prediction of continuous actions for deep RL. CoRR **abs/1705.05035** (2017). URL `http://arxiv.org/abs/1705.05035`
13. Moré, J.: The levenberg-marquardt algorithm: Implementation and theory. In: G. Watson (ed.) Numerical Analysis, *Lecture Notes in Mathematics*, vol. 630, pp. 105–116. Springer Berlin Heidelberg (1978). DOI 10.1007/BFb0067700. URL `http://dx.doi.org/10.1007/BFb0067700`
14. Peters, J., Schaal, S.: Natural actor-critic. Neurocomputing **71**(7-9), 1180–1190 (2008)
15. Peters, J., Vijayakumar, S., Schaal, S.: Reinforcement learning for humanoid robotics. In: IEEE-RAS International Conference on Humanoid Robots (Humanoids2003) (2003). URL `https://www.ias.informatik.tu-darmstadt.de/uploads/Team/JanPeters/peters-ICHR2003.pdf`
16. Schulman, J., Levine, S., Abbeel, P., Jordan, M., Moritz, P.: Trust region policy optimization. In: F. Bach, D. Blei (eds.) Proceedings of the 32nd International Conference on Machine Learning, *Proceedings of Machine Learning Research*, vol. 37, pp. 1889–1897. PMLR, Lille, France (2015). URL `http://proceedings.mlr.press/v37/schulman15.html`
17. Schulman, J., Moritz, P., Levine, S., Jordan, M.I., Abbeel, P.: High-dimensional continuous control using generalized advantage estimation. CoRR **abs/1506.02438** (2015). URL `http://arxiv.org/abs/1506.02438`
18. Sutton, R.S.: Learning to predict by the methods of temporal differences. Machine Learning **3**(1), 9–44 (1988). DOI 10.1007/BF00115009. URL `https://doi.org/10.1007/BF00115009`
19. Sutton, R.S., McAllester, D., Singh, S., Mansour, Y.: Policy gradient methods for reinforcement learning with function approximation. In: Proceedings of the 12th International Conference on Neural Information Processing Systems, NIPS'99, pp. 1057–1063. MIT Press, Cambridge, MA, USA (1999). URL `http://dl.acm.org/citation.cfm?id=3009657.3009806`
20. Williams, P.: Using neural networks to model conditional multivariate densities. Neural computation **8**, 843–54 (1996). DOI 10.1162/neco.1996.8.4.843
21. Williams, R.J.: Simple statistical gradient-following algorithms for connectionist reinforcement learning. In: Machine Learning, pp. 229–256 (1992)
22. Zhang, R., Chen, C., Li, C., Carin, L.: Policy optimization as wasserstein gradient flows. CoRR **abs/1808.03030** (2018). URL `http://arxiv.org/abs/1808.03030`