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

Natural Actor-Critic: Components and Extensions

Maximilian A. Gehrke ·
Yannik P. Frisch · **Tabea A. Wilke**

Received: date / Accepted: date

Abstract In this paper we describe the natural actor-critic approach and provide an extensive overview of the current research. This includes a basic description of the natural gradient, an introduction to actor-critic approaches and comparisons between existing natural actor-critic modifications.

Keywords Natural Gradient · Advantage Function · Actor-Critic · NAC

1 Introduction

Natural actor-critic (NAC) methods [21] have been very successful in the last two decades. They could be applied to various fields, including traffic optimization [22], dialog systems [13] and high dimensional control tasks [17, 19, 20]. The NAC is an actor-critic policy gradient method (PGM), which optimizes a policy using gradient ascent.

PGMs represent a policy by using differentiable function approximation. They optimize a scalar performance measure J, called objective function, by repeatedly estimating its gradient w.r.t. the policy parameters and updating the policy parameters a proportion in its direction. PGMs have several advantages. In comparison to value-based algorithms, policy gradient methods have very good sample complexity guarantees [16] and convergence properties. In addition, PGMs are model-free, can learn stochastic policies and are effective in high-dimensional or continuous action spaces. Further, with PGM we can introduce a prior on the policy, we can converge to a deterministic policy and

Maximilian A. Gehrke
E-mail: maximilian_alexander.gehrke@stud.tu-darmstadt.de

Yannik P. Frisch
E-mail: yannik_phil.frisch@stud.tu-darmstadt.de

Tabea A. Wilke
E-mail: tabeaalina.wilke@stud.tu-darmstadt.de

do not have to choose a suboptimal action for exploration purposes and we can choose our actions stochastically. However, PGMs are typically inefficient, have high variance and typically converge to a local rather than a global optimum. It is also necessary, that the policy is differentiable w.r.t its parameters, which is not always the case.

Actor-critic methods are special cases of PGMs. They approximate a value function (typically a state-action value function), which helps to approximate the policy. This means that we have to estimate two sets of parameters: the parameters of the policy (*actor*) and the parameters of the value function (*critic*). The actor tells the agent, which actions to execute. The agent executes the action in the environment and the environment returns an observation. The critic rates the observations and updates its own parameters. Immediately afterwards or if a specific criterion is met, the actor updates its parameters w.r.t the critic.

The NAC utilizes the natural gradient in an actor-critic environment to optimize a policy. How this works and possible modifications to the NAC will be the scope of this paper. We start by setting up some preliminaries in section 2, before we introduce and discuss the natural gradient in section 3. The natural actor-critic algorithm is presented in section 4 and modifications to the NAC in section 5. Finally, we close with a discussion in section 6.

2 Preliminaries

We consider a standard reinforcement learning framework, in which a learning agent interacts with a Markov Decision Process (MDP) [11, 26]. For each discrete time step $t \in \{0, 1, 2, ...\}$, the state, action and reward is denoted as $s_t \in S$, $a_t \in A$ and $r_{t+1} \in R \subset \mathbb{R}$ respectively. The dynamics of the environment are described by the state-transition probabilities $p(s|s', a) = \Pr\{S_t = s'|S_{t-1} = s, A_{t-1} = a\}$ and the expected immediate rewards $r(s, a) = \mathbb{E}[R_t|S_{t-1} = s, A_{t-1} = a]$, for all $s, s' \in S, a \in A$. The agent's behavior at each time step t is specified by a policy $\pi_\theta(a|s) = \Pr\{A_t = a|S_t = s, \theta\}$, where θ denotes the parameters of the policy.

We assume that π is differentiable w.r.t. its parameters, so that $\frac{\partial \pi(a|s)}{\partial \theta}$ exists and we can estimate the gradient of the objective function $J(\theta)$ by applying the policy gradient theorem [27]

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a|s) Q^{\pi_\theta}(s, a)], \tag{1}$$

where $Q^{\pi_\theta}(s, a)$ denotes an action-value function. One of the most basic policy gradient algorithms, *REINFORCE* [31], estimates the action-value function Q^{π_θ} by using the expected discounted return (also known as Monte-Carlo return), $Q^{\pi_\theta}(s, a) \approx G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$, where $\gamma \in [0, 1]$ denotes a discount factor. With the gradient of the objective function $\nabla_\theta J(\theta)$ and a learning rate $\alpha \in [0, 1]$, policy gradient methods recursively update the parameters of the policy, $\theta_{t+1} = \theta_t + \alpha \nabla_\theta J(\theta)$. After a finite number of steps, the parameters converge to a local optimum.

Instead of using a Monte-Carlo estimate directly, actor-critic methods employ the critic to model the action-value function with a function approximation $Q^{\pi_\theta}(s, a) \approx Q_w(s, a)$ [27], where w denotes the parameters of the critic, which need to be optimized. In addition, the introduction of a baseline $B(s, a)$, reduces the variance of the action-value function estimate and accelerates learning [26]: $\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a|s) Q_w(s, a) - B(s, a)]$. A good baseline with minimal variance is the value function. Subtracting the value function from the action-value function yields the advantage function $A(s, a) = Q(s, a) - V(s)$. The critic is able to directly estimate the advantage function $A_w(s, a)$:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a|s) A_w(s, a)]. \tag{2}$$

Equation 2 will help us understand how the NAC algorithm in section 4 works. Beforehand, we need to get an understanding of the natural gradient.

3 Natural Gradient

The natural gradient was first introduced by Amari in 1998 [2]. The difference between the natural gradient and the ordinary vanilla gradient is the direction it points to. The ordinary vanilla gradient only points to the steepest direction, if the parameter space is orthonormal and has a Euclidean character. The natural gradient, however, points to the steepest direction of a Riemann parameter space (e.g. neural networks [2]).

A Riemann parameter space is a differentiable manifold, where an inner product $< \cdot, \cdot >$ exists for each tangent space. For two tangent vectors $\mathbf{u}$ and $\mathbf{v}$, the inner product $< \mathbf{u}, \mathbf{v} >$ yields a real number. This makes it possible to define notions such as length, areas, angles or volumes. To calculate the gradient, we need to be able to calculate the squared length of a small incremental vector $d\mathbf{w}$ connecting a point $\mathbf{w}$ and $\mathbf{w} + d\mathbf{w}$. Equation 3 shows the formula for Riemann spaces (on the left) and Euclidean spaces (on the right):

$$|d\mathbf{w}|^2 = \sum_{i,j} g_{ij}(\mathbf{w}) dw_i dw_j \ , \ |d\mathbf{w}|^2 = \sum_{i=1}^{n} (dw_i)^2, \tag{3}$$

where $g_{ij}(\mathbf{w})$ is a function, enabling us to create a measure of distance. It can be written as an $n \times n$ matrix $G = (g_{ij})$, called Riemannian metric tensor, and reduces to the unit matrix I in the case of a Euclidean orthonormal parameter space. Therefore, the Riemannian distance measure is a generalization of the Euclidean orthonormal measure of distance [9, 2]. We can utilize the Riemannian metric tensor to construct a gradient which points in the steepest direction of a Riemannian space:

$$\widetilde{\nabla}_\theta J(\theta) = G^{-1} \nabla_\theta J(\theta). \tag{4}$$

$\widetilde{\nabla}_\theta$ is the natural gradient w.r.t the parameters θ. Learning should be carried out with a gradient descent like update rule: $\theta_{t+1} = \theta_t + \alpha \widetilde{\nabla}_\theta J(\theta)$, where α denotes a learning rate as usual. In the special case that the parameter space is Euclidean and the coordinate system orthonormal, the conventional gradient equals the natural gradient: $\widetilde{\nabla}_\theta J(\theta) = \nabla_\theta$. If the Fisher information matrix (FIM) exists, it has been shown that we can use it as the Riemannian metric tensor in equation 4 to get the natural gradient [19, 1]. The FIM of a policy is defined as:

$$F_\theta = \mathbb{E}_{s,a} \left[\nabla_\theta \log \pi_\theta(a|s)^T \nabla_\theta \log \pi_\theta(a|s) \right] . \tag{5}$$

If we look at the problem from a different angle, we can see why this approach is unique. Policy gradient methods change the parameters of the policy in order to maximize an objective function $J(\theta)$. This can be done by taking the vanilla gradient (equation 6). However, the vanilla gradient has the downfall that in flat regions of the parameter space, the algorithm moves very slowly, whereas in steep regions the algorithm moves very fast and may even shoot beyond the local optimum. The reason for this is that for every parametrization of π_θ the gradient is different. This is why we introduce the Kullback-Leibler divergence, a measure of distance between two distributions, which can be approximated by the second-order Taylor expansion (equation 7). We constrain the Kullback-Leibler divergence to be less than a fixed value ϵ, so that for each update the parameters of π_θ change exactly for a given distance in parameter space. Equation 6 and 7 together form an optimization problem:

$$\max_{\delta\theta} J(\theta + \delta\theta) \approx J(\theta) + \delta\theta^T \nabla_\theta J(\theta) \tag{6}$$

$$\text{s.t. } \epsilon = D_{KL}(\pi_\theta || \pi_{\theta+\delta\theta}) \approx \tfrac{1}{2} \delta\theta^T F_\theta \delta\theta. \tag{7}$$

The solution of the optimization problem yields equation 4, applied with the FIM. Now we can see why the natural gradient is so unique: it is invariant to parametrization [18, 19]. In addition, the natural gradient (NG) has some other important properties, which we would like to mention:

- **Online Learning:** The NG can be used online and therefore can learn from incomplete sequences and reduce the variance of the action-value function estimation [18, 19].
- **1st order method:** The NG is a first order method, but implements second order advantages [18]. This is especially relevant for problems, where the cost function is only accessible indirectly [8].
- **Better & faster convergence:** In many cases, the NG converges faster than vanilla gradient methods and avoids getting stuck in plateaus [1, 25].
- **Drawbacks:** The Riemannian metric tensor needs to be nonsingular and invertible. This is not always the case and even if it is the case, the inversion of a matrix is very costly. In addition, by applying the NG, the policy variance might reduce to zero, which in turn reduces the exploration to zero. However, exploration is needed to find an optimal policy.

4 Natural Actor-Critic

In this paper, we focus on the trajectory-based formulation of the *Natural Actor-Critic* (NAC) algorithm [21], called episodic NAC (eNAC, algorithm 1).

Algorithm 1 Episodic Natural Actor-Critic (eNAC)

```
Require: Parameterized policy $\pi_\theta(a|s)$ and its derivative $\nabla_\theta \log \pi_\theta(a|s)$
         with initial parameters $\theta = \theta_0$.
for $u = 1, 2, 3, \ldots$ do
    for $e = 1, 2, 3, \ldots$ do
        Execute roll-out: Draw initial state $s_0 \sim p(s_0)$
        for $t = 1, 2, 3, \ldots, N$ do
            Draw action $a_t \sim \pi_{\theta_t}(a_t|s_t)$, observe next state $s_{t+1} \sim p(s_{t+1}|s_t, a_t)$
            and reward $r_{t+1} = r(s_t, a_t)$.
        end for
    end for
    Critic Evaluation (repeat for each sampled trajectory): Determine compatible
    function approximation of advantage function $A(s,a) \approx A_{w_t}(s,a)$.
    Determine basis functions: $\Phi_e = \left[\sum_{t=0}^{T} \gamma^t \nabla_\theta \log \pi_\theta(a_t|s_t)^T, 1\right]^T$,
    reward statistics: $R_e = \sum_{t=0}^{T} \gamma^t r_t$ and solve $\begin{bmatrix} w_e \\ J \end{bmatrix} = (\Phi_e^T \Phi_e)^{-1} \Phi_e^T R_e$.
    Update critic parameters: $w_{t+1} = w_t + \beta w_e$.
    Actor Update: When the natural gradient is converged, $\measuredangle(w_{t+1}, w_t) \leq \epsilon$, update
    the policy parameters: $\theta_{t+1} = \theta_t + \alpha w_{t+1}$.
end for
```

The eNAC algorithm has a fixed amount of updates u and a fixed amount of steps the agent executes every update. Therefore, if a trajectory e has reached a terminal state before the agent executed all its steps, the algorithm samples a new trajectory. This repeats until the maximum number of steps is met and the current trajectory is interrupted. During this process, all states, actions and rewards are stored for each trajectory.

Afterwards, we estimate w_e by determining compatible function approximations, basis functions and reward statistics for the samples of a single trajectory e and solve a linear system of equation. w_t is then updated by adding a proportion of w_e. We repeat this process for all trajectories encountered during the update. If the angle between w_{t+1} and w is smaller than some fixed value ϵ, we update the policy parameters θ a proportion in the direction of w_{t+1}.

Critic Update: For the critic, we use a compatible function approximation of the advantage function. The use of a compatible function approximation $A_w(s,a)$ to estimate $A(s,a)$ [27] is a key aspect of the eNAC algorithm. By definition, a compatible function approximation has the property that its gradient can be expressed in terms of the policy. This means, that we can express the advantage function by taking the derivative w.r.t. the policy and multiplying it by w:

$$\nabla_w A_w(s,a) = \nabla_\theta \log \pi_\theta(a|s) \tag{8}$$
$$A_w(s,a) = \nabla_\theta \log \pi_\theta(a|s) w. \tag{9}$$

To estimate the parameters w of the advantage function, we notice that the discounted sum of advantages can be written in terms of the expected reward and value function

$$\sum_{t=0}^{N} \gamma^t A(s_t,a_t) = \sum_{t=0}^{N} \gamma^i r(s_t,a_t) + \gamma^N V(S_{N+1}) - V(S_0), \tag{10}$$

where N is the number of steps executed in a trajectory [21]. If we assume $\gamma \neq 1$, we can remove the term $\gamma^N V(S_{N+1})$, because in the limit the term becomes zero ($\lim_{N\to\infty} \gamma^N = 0$). Additionally, if we assume that we always start at the same start state S_0, we can write $V(S_0)$ as the cost function $J(\theta)$. Inserting the approximated advantage function (equation 9) for $A(s_t,a_t)$ and bringing the cost function $J(\theta)$ to the left-hand side, yields:

$$\sum_{i=0}^{N} \gamma^i \nabla_\theta \log \pi_\theta(a_i|s_i)^T \cdot w + J(\theta) = \sum_{i=0}^{N} \gamma^i r(s_i,a_i). \tag{11}$$

This is exactly the equation, which we solve in algorithm 1 by taking the left pseudoinverse. Besides the parameter vector w, we receive the cost function $J(\theta)$ as a side product.

Actor Update: The reason why we update the policy parameters in the direction of the critic's parameters is the utilization of a compatible function approximation to estimate the advantage function (equation 9). With this the natural policy gradient from equation 2 simplifies in the following way:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(a|s) A_w(s,a)\right] \tag{12}$$
$$= \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(a|s) \nabla_\theta \log \pi_\theta(a|s)^T w\right] = G_\theta w \tag{13}$$
$$\widetilde{\nabla}_\theta J(\theta) = w \tag{14}$$

In the next section we present several modifications to the NAC algorithm, which among others improve the accuracy and performance of the critic's estimation, data efficiency, learning stability and matrix inversion.

5 NAC Modifications

Least Squares: Besides the episodic NAC the original NAC paper [21] featured another approach: NAC using LSTD-Q(λ) [14, 6]. The main difference is the estimation of the critic's parameters. LSTD-Q(λ) uses least squares temporal difference learning [5] to estimate the parameters of the critic after every step taken in the environment. The algorithm uses eligibility traces [26] and two linear functions: $A_w(s,a) = \nabla_\theta \log \pi_\theta(a|s)^T w$ and $V_v(s) = \phi(s)v$. The latter is an approximation of the value function, needed to update the critic by solving a linear set of equations induced by the least squares error between the observation and the value function approximation.

Recursive Least Squares: The "RLS-based natural actor-critic algorithm" equips the LSTD-Q NAC algorithm with a recursive update rule for the parameters of the critic [17]. The old parameter values are reused during the current update, which increases efficiency [33].

Fitted NAC & Importance Sampling: Fitted natural actor-critic (FNAC) is a fitted version of the natural actor-critic algorithm [15]. It employs a memory D, which is filled by sampling data from the environment. Once filled, the least squares NAC algorithm is executed as usual. Normally, after a policy update, we would need to sample D again with the improved policy. However, implementing importance sampling (IS) [26] avoids the re-sampling. In addition to the current policy parameters θ, IS saves the policy parameters θ^-, which were used to sample the memory D. Hence, the memory D is independent of the current learning policy. Every time we evaluate the critic, we multiply our estimation by the importance weights, $\frac{\pi_\theta(a|s)}{\pi_{\theta^-}(a|s)}$, to estimate the proportion we need to change the critic's parameters. This approach is extremely data efficient and brings fundamental advantages in situations, where collecting data is costly or time-consuming. Additionally, FNAC makes use of regression methods to update the critic's parameters, which allow the use of a general function approximation for the value function instead of a linear approximation. This positively impacts the accuracy of the critic's estimation.

Incremental NAC (INAC): The incremental NAC algorithm combines linear function approximation and bootstrapping [4]. It reuses existing approaches, namely temporal difference learning [26] and two-timescale stochastic approximation [3]. Bhatnagar et al. provide three new natural gradient algorithms and prove local convergence. The main feature of the algorithms is the incremental estimation of the policy and the incremental update of the gradient. The policy is changed every time step, and, in comparison to Peters et al., the policy gradient is not reset every update, but saved and reused to calculate the gradient of the next iteration. These improvements facilitate the application to large-scale reinforcement learning problems, decreases computation time and make the algorithm more efficient than conventional actor-critic methods. Further, one of the algorithms can be executed without explicitly computing the inverse Fischer information matrix, which, so the authors, leads to even faster convergence.

Implicit Incremental NAC (I2NAC): INAC algorithms suffer from a difficult to tune step size and an unstable, sometimes divergent, estimation of the natural gradient. The implicit incremental NAC [12] uses the ideas of implicit stochastic gradient descent [30] and implicit temporal differences [28] to overcome these difficulties. The change between INAC and I2NAC is a scalar weight, which is applied when updating the critic. The algorithm uses eligibility traces and a new hyper-parameter γ to compute the weight. This stabilizes learning and empirical results show less divergence.

Regularization on NAC: Even if we find the inverse of the Riemannian metric tensor G (equation 4), it can be ill-defined. An example for this are extremely small eigenvalues, which appear due to noise in the data. These eigenvalues will become extremely large if we take the inverse of G and thus the parameters belonging to the eigenvalues will get a lot of credibility, which they should not have and which falsifies the inverse.

There have been several approaches to introduce a regularization term [25, 32]. Regularizing the matrix inverse can be done by a technique called "stochastic robust approximation" [7], where G^{-1} is replaced by $G_{\text{reg}}^{-1} = \left(G^T G + \epsilon I\right)^{-1} G^T$ and ϵ denotes a small constant (e.g 0.01).

Another idea is the application of ridge regression [10], which has a build in regularizer. We can calculate $\widetilde{\nabla}_\theta J(\theta)$ by solving the linear equation $G(\theta)\widetilde{\nabla}_\theta J(\theta) = \nabla_\theta J(\theta)$ in the direction of $\widetilde{\nabla}_\theta J(\theta)$.

6 Discussion

In this paper, we described the natural gradient, the natural actor-critic algorithm and NAC modifications, which have been applied in the last years. NAC is a model-free state-of-the-art algorithm that can be applied to continuous action spaces. It is parametrization invariant, has been reported to converge faster than vanilla gradient methods and can jump out of plateaus, which is why we expect to see more use of NACs in the future.

Disadvantages are the inversion of a matrix and the extinction of variance and therefore exploration. For the first, people may need to come up with faster solution to invert matrices or even how to avoid the estimation. The latter has already been tackled by some algorithms, namely TRPO [23] and PPO [24]. What is more prevailing, however, is a study claiming that NACs exhibit a bias [29]. Further research is needed to shed light on this issue.

Most of the modifications presented in section 5 need evaluation on real-world problems to assess their ultimate utility; many of the modifications are only applied to specific environments. Broader research will be necessary to integrate different approaches (eligibility traces, least-squares methods, online implementations) and survey which approach works best in which situations.

In conclusion, NACs work very well for many standard reinforcement learning problems where policy gradient methods are employable. We even encourage the application to POMDPs, where "The Natural Actor and Belief Critic" [13] demonstrated promising first successes.

References

1. Shun-Ichi Amari. Natural gradient works efficiently in learning. *Neural computation*, 10(2):251–276, 1998.
2. Shun-Ichi Amari and Scott C Douglas. Why natural gradient? In *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP'98 (Cat. No. 98CH36181)*, volume 2, pages 1213–1216. IEEE, 1998.
3. Shalabh Bhatnagar and Vivek S Borkar. A two timescale stochastic approximation scheme for simulation-based parametric optimization. *Probability in the Engineering and Informational Sciences*, 12(4):519–531, 1998.
4. Shalabh Bhatnagar, Mohammad Ghavamzadeh, Mark Lee, and Richard S Sutton. Incremental natural actor-critic algorithms. In *Advances in neural information processing systems*, pages 105–112, 2008.
5. Justin A Boyan. Least-squares temporal difference learning. In *ICML*, pages 49–56, 1999.
6. Justin A Boyan. Technical update: Least-squares temporal difference learning. *Machine learning*, 49(2-3):233–246, 2002.
7. Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.
8. Guillaume Desjardins, Razvan Pascanu, Aaron Courville, and Yoshua Bengio. Metric-free natural gradient for joint-training of boltzmann machines. *arXiv preprint arXiv:1301.3545*, 2013.
9. Simon S Haykin, Simon S Haykin, Simon S Haykin, Kanada Elektroingenieur, and Simon S Haykin. *Neural networks and learning machines*, volume 3. Pearson Upper Saddle River, 2009.
10. Arthur E Hoerl and Robert W Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.
11. Ronald A Howard. Dynamic programming and markov processes. 1960.
12. Ryo Iwaki and Minoru Asada. Implicit incremental natural actor critic algorithm. *Neural Networks*, 109:103–112, 2019.
13. Filip Jurčíček, Blaise Thomson, and Steve Young. Natural actor and belief critic: Reinforcement algorithm for learning parameters of dialogue systems modelled as pomdps. *ACM Transactions on Speech and Language Processing (TSLP)*, 7(3):6, 2011.
14. Michail G Lagoudakis and Ronald Parr. Least-squares policy iteration. *Journal of machine learning research*, 4(Dec):1107–1149, 2003.
15. Francisco S Melo and Manuel Lopes. Fitted natural actor-critic: A new algorithm for continuous state-action mdps. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 66–81. Springer, 2008.
16. Arkadi Nemirovski. Efficient methods in convex programming. 2005.
17. Jooyoung Park, Jongho Kim, and Daesung Kang. An rls-based natural actor-critic algorithm for locomotion of a two-linked robot arm. In *International Conference on Computational and Information Science*, pages 65–72. Springer, 2005.

18. Razvan Pascanu and Yoshua Bengio. Revisiting natural gradient for deep networks. *arXiv preprint arXiv:1301.3584*, 2013.
19. Jan Peters and Stefan Schaal. Natural actor-critic. *Neurocomputing*, 71(7-9):1180–1190, 2008.
20. Jan Peters, Sethu Vijayakumar, and Stefan Schaal. Reinforcement learning for humanoid robotics. In *Proceedings of the third IEEE-RAS international conference on humanoid robots*, pages 1–20, 2003.
21. Jan Peters, Sethu Vijayakumar, and Stefan Schaal. Natural actor-critic. In *European Conference on Machine Learning*, pages 280–291. Springer, 2005.
22. Silvia Richter, Douglas Aberdeen, and Jin Yu. Natural actor-critic for road traffic optimisation. In *Advances in neural information processing systems*, pages 1169–1176, 2007.
23. John Schulman, Sergey Levine, Pieter Abbeel, Michael I Jordan, and Philipp Moritz. Trust region policy optimization. In *Icml*, volume 37, pages 1889–1897, 2015.
24. John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.
25. Jascha Sohl-Dickstein. The natural gradient by analogy to signal whitening, and recipes and tricks for its use. *arXiv preprint arXiv:1205.1828*, 2012.
26. Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.
27. Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pages 1057–1063, 2000.
28. Aviv Tamar, Panos Toulis, Shie Mannor, and Edoardo M Airoldi. Implicit temporal differences. *arXiv preprint arXiv:1412.6734*, 2014.
29. Philip Thomas. Bias in natural actor-critic algorithms. In *International Conference on Machine Learning*, pages 441–448, 2014.
30. Panagiotis Toulis, Edoardo Airoldi, and Jason Rennie. Statistical analysis of stochastic gradient methods for generalized linear models. In *International Conference on Machine Learning*, pages 667–675, 2014.
31. Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.
32. Andreas Witsch, Roland Reichle, Kurt Geihs, Sascha Lange, and Martin Riedmiller. Enhancing the episodic natural actor-critic algorithm by a regularisation term to stabilize learning of control structures. In *2011 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*, pages 156–163. IEEE, 2011.
33. Xin Xu, Han-gen He, and Dewen Hu. Efficient reinforcement learning using recursive least-squares methods. *Journal of Artificial Intelligence Research*, 16:259–292, 2002.