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

Reviewing and improving Natural Actor Critic methods

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 about the current research. This includes a basic description of the natural gradient, actor critic approaches and comparisons between existing extensions. Additionally, we improve the episodic Natural Actor Critic algorithm by applying it with two neural networks instead of basis functions.

Keywords Natural Gradient · Neural Networks · Advantage Function · Actor Critic · NAC

Maximilian A. Gehrke
TU Darmstadt, Germany
E-mail: maximilian_alexander.gehrke@stud.tu-darmstadt.de

Yannik P. Frisch
TU Darmstadt, Germany
E-mail: yannik_phil.frisch@stud.tu-darmstadt.de

Tabea A. Wilke
TU Darmstadt, Germany
E-mail: tabeaalina.wilke@stud.tu-darmstadt.de

1 Introduction

Policy gradient methods have dominated the field of reinforcement learning in the last years [LOT OF CITES]. These methods represent the policy using differentiable function approximation. They optimize an objective function $J(\theta)$ by repeatedly estimating the gradient of the approximated return w.r.t. the parameters θ of the policy and updating the parameters in that direction by stochastic gradient descent.

2 Preliminaries

We consider a standard reinforcement learning framework, in which a learning agent interacts with a Markov Decision Process (MDP) [9, 22]. 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 $P^a_{ss'} = \Pr\{S_t = s' | S_{t-1} = s, A_{t-1} = a\}$ and the expected immediate rewards $R^a_s = \mathbb{E}[R_t | S_{t-1} = s, A_{t-1} = a]$, 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. it's 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 [23]

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

where $Q^{\pi_\theta}(s, a)$ denotes an action-value function. One of the most basic policy gradient algorithms, *REINFORCE* [25], estimates the action-value function Q^{π_θ} by using the expected discounted return, also known as Monte-Carlo return

$$Q^{\pi_\theta}(s, a) \approx G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}, \tag{2}$$

where γ is a discount factor with $\gamma \in [0, 1]$. Policy gradient methods use the gradient of the objective function $J(\theta)$ and a learning rate $\alpha \in [0, 1]$ to recursively update the parameters of $\pi_\theta(a|s)$, $\theta_{t+1} = \theta_t + \alpha \nabla_\theta J(\theta)$, and find a local optimum.

However, instead of using a Monte-Carlo estimate directly, Actor-Critic methods model the action-value function with a function approximator $Q^{\pi_\theta}(s, a) \approx Q_w(s, a)$ [23]. $Q_w(s, a)$ is called the critic and introduces a second set of parameters, w, which need to be optimized; $\pi_\theta(a|s)$ is called the actor. By introducing a baseline $B(s, a)$, we can reduce the variance of the action-value function estimate and accelerate learning [22]:

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

A good baseline with minimal variance is the value function. Subtracting the value function from the action-value function yields the Advantage function $A(s,a) = Q(s,a) - V(s)$. However, the critic can directly estimate the advantage function $A_w(s,a)$ for computing the gradient:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a|s) A_w(s,a)]. \tag{4}$$

Optimzing the objective function with vanilla gradient descent is sensitive to parametrization and can be inefficient. One could instead use the *natural gradient*, which is described in the next section.

3 Natural Gradient

[1] 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 does only point to the steepest direction, if the parameter space has an Euclidean character [x]. The natural gradient, however, points to the steepest direction of a Riemann parameter space, which is a generalization of the Euclidean parameter space [7, 2]. This is especially convenient, if our function approximator exhibits a Riemannian character, e.g. neural networks[2].

The natural gradient utilizes the $n \times n$ matrix $G = (g_{ij})$, called Riemannian metric tensor, which in general depend on θ. It is defined as

$$\widetilde{\nabla}_\theta J(\theta) = G^{-1} \nabla_\theta J(\theta) \tag{5}$$

where $\widetilde{\nabla}_\theta$ is the natural gradient w.r.t the parameters θ. Learning should be carried out with a gradient descent like update rule: $\theta_{t+1} = \theta_t + \alpha \widetilde{\nabla}_\theta J(\theta)$. In the special case that the parameter space is Euclidean and the coordinate system is orthonormal, the conventional gradient equals the natural gradient: $\widetilde{\nabla}_\theta J(\theta) = \nabla_\theta$.

The natural gradient is the steepest ascent direction of our performance object with respect to any metric. Well, but how do we calculate the matrix G? The Hessian H at x_0 (which exists since the first derivative vanishes) is the purely covariant form of the Riemann curvature tensor. To estimate the Hessian H, we can use the Fisher information metric

$$F_\theta = \mathbb{E}_{s\sim\rho^\pi, a\sim\pi_\theta}\left[\nabla_\theta log \pi_\theta(a|s)^T \nabla_\theta log \pi_\theta(a|s)\right] \tag{6}$$

For deterministic policies:

$$M_\mu(\theta) = E_{s\sim\rho^\mu}[\nabla_\theta \mu_\theta(s) \nabla_\theta \mu_\theta(s)^T w] \tag{7}$$

$\rightarrow$ Limiting case of the Fisher information metric: policy variance reduced to zero.
Combining DPG theorem with compatible function approximation gives

$$\nabla_\theta J(\mu_\theta) = E_{s\sim\rho^\mu}[\nabla_\theta \mu_\theta(s) \nabla_\theta \mu_\theta(s)^T w] \tag{8}$$

so steepest ascent direction reduces to

$$M_\mu(\theta)^{-1} \nabla_\theta J_\beta(\mu_\theta) = w \tag{9}$$

Natural gradient algorithms have been applied successfully in various fields such as road traffic optimization [20], robotic control tasks [11], motor primitive learning [17] and locomotion of a two-linked robot arm [15]. Using the natural gradient has several advantages and properties, which are listed in the next section.

4 Properties of the natural gradient

In the following we present several properties of using the natural gradient for optimizing a reinforcement learning loss function.

4.1 Online Learning

The natural gradient can be used online. This means that we can learn from incomplete sequences and by this reduce the variance of our estimation of the return or the advantage function. It is also possible to use an offline, episodic variant of the natural gradient which can be seen in section ??. With both variations at hand, one can make the decision regarding the structure of our problem [16, 18].

4.2 1st order method

The natural gradient is a first order method, but implements many second order advantages [16]. This is especially relevant for problems, where the cost function is accessible indirectly. For example Deep Boltzmann Machines exhibit this characteristics and recent studies showed possible solutions with natural gradients [6] whereas second order methods are hard to apply.

4.3 Parameterization invariant

The natural gradient is invariant regarding parameterizations. This is a result from the KL-divergence constraint we apply, which measures the changes of our probability density estimation regardless how it was parameterized [16]. This is comparable to signal noise whitening [21].

4.4 Faster convergence

In many cases natural gradient algorithms converge faster than vanilla gradient algorithms [21, 2]. This is due to the fact that a metric is used which removes the dependencies and differences in scaling between the parameters. This means that these are perpendicular to each other and their space can be explored more efficiently [21].

4.5 Better convergence properties

Using the natural gradient instead of the vanilla gradient avoids getting stuck in local optima during optimization [2].

4.6 Sample efficiency

Natural actor critic has much better sample complexity guarantees than gradient-free algorithms, due to the fact that it is a gradient method [14].

4.7 More properties

Look at Berkley slides

4.8 Drawbacks

The matrix W is nonsingular (i.e., invertible).

4.9 NN and learning machines

1. The algorithm is computationally efficient, since it avoids the need for inverting the demixing matrix W. 2. The convergence rate of the algorithm is relatively fast. 3. The algorithm lends itself to implementation in the form of an adaptive neural system. 4. Being a stochastic gradient algorithm, the algorithm has a built-in capability to track statistical variations of a nonstationary environment.

We will present an application of using the natural gradient together with an actor critic method in the next section.

5 Natural Actor Critic

In this section we describe the *Natural Actor Critic* (NAC) algorithm [19]. We focus on the trajectory based formulation, called episodic NAC, and present the pseudo code in algorithm 1.

In episodic NAC, we have a fixed amount of updates u and a fixed amount of steps the agent executes in the environment every update. Therefore, if a trajectory e has reached a terminal state before the agent executed all it's 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 we see, all actions we take and all rewards we get are stored for each trajectory.

After sampling, we perform the critic evaluation. We determine the compatible function approximation, the basis functions and the reward statistics for the samples of a single episode and solve a linear equation system to get w_e. We update w_t by adding w_e multiplied by a learning rate $\beta \in [0, 1]$. We repeat this process for all trajectories encountered during the update. Then, we check if the angle between w_{t+1} and w is smaller than some fixed value ϵ and if so, we update the policy parameters θ by adding w_{t+1} multiplied by a learning rate $\alpha \in [0, 1]$.

Algorithm 1 Episodic Natural Actor Critic (eNAC)

```
Require: Parameterized policy $\pi_\theta(a|s)$ and it's derivative $\nabla_\theta \log \pi_\theta(a|s)$
         with initial parameters $\theta = \theta_0$.
 1: for $u = 1, 2, 3, \ldots$ do
 2:   for $e = 1, 2, 3, \ldots$ do
 3:     Execute roll-out: Draw initial state $s_0 \sim p(s_0)$
 4:     for $t = 1, 2, 3, \ldots, N$ do
 5:       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 $r_{t+1} = r(s_t, a_t)$.
 6:     end for
 7:   end for
 8:   Critic Evaluation (repeat for each sampled trajectory): Determine compatible
      function approximation of advantage function $A(s, a) \approx A_{w_t}(s, a)$.
 9:   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: $w_{t+1} = w_t + \beta w_e$.
10:   Actor Update: When the natural gradient is converged, $\measuredangle(w_{t+1}, w_t) \leq \epsilon$, update
      the policy parameters: $\theta_{t+1} = \theta_t + \alpha w_{t+1}$.
11: end for
```

Our cost function is the expected return $\mathbb{E}[G_t]$, which can be written as the discounted sum of advantages, which in return can be written in terms of the expected reward and value function [19]

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

where N is the number of steps executed in a trajectory. 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 in the same start state S_0, we can write $V(S_0)$ as our cost function $J(\theta)$:

$$\sum_{t=0}^{N} \gamma^t A(s_t, a_t) = \sum_{t=0}^{N} \gamma^t r(s_t, a_t) - J(\theta). \tag{11}$$

One of the key aspects of the NAC algorithm is the use of a compatible function approximation $A_w(s,a)$ to estimate $A(s,a)$ [23], which by definition has the property that it's gradient can be expressed in terms of the policy. This also means, that we can express the advantage function by taking the derivative w.r.t. the policy and multiplying it by w:

$$\nabla_w A_w(s,a) = \nabla_\theta \log \pi_\theta(s|a) \tag{12}$$

$$A_w(s,a) = \nabla_\theta \log \pi_\theta(s|a) w. \tag{13}$$

Inserting this and bringing the cost function $J(\theta)$ to the left hand side:

$$\sum_{i=0}^{N} \gamma^i \nabla_\theta \log \pi_\theta(a_i|s_i)^T \cdot w + 1 \cdot J = \sum_{i=0}^{N} \gamma^i r(s_i, a_i) \tag{14}$$

This is exactly the equation, which we solve in algorithm 1 by taking the left pseudo inverse to update the critics parameters w.

The update of the policy parameters in direction of the critic parameters can also be explained by the compatible function approximation (equation 13). With this the natural policy gradient [REF: EQUATION] simplifies:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s|a) A_w(s,a)\right] \tag{15}$$

$$= \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s|a) \nabla_\theta \log \pi_\theta(s|a)^T w\right] \tag{16}$$

$$= G_\theta w \tag{17}$$

$$\widetilde{\nabla}_\theta J(\theta) = w \tag{18}$$

We show several modifications to the algorithm in the next section and discuss it's advantages and drawbacks.

6 Modifications

In this section we list several extensions to the NAC algorithm.

6.1 LSTD-Q NAC

Peters did not only specify the episodic NAC algorithm in his original paper [19], but also an NAC algorithm using LSTD-Q(λ) [12, 4]. LSTD-Q(λ) uses Least Squares Temporal Difference learning to learn the weighted least squares fixed-point approximation of the state-action value function of a fixed policy π.
but the LSTD-Q approach is analog with the main difference that we have one update per action instead of one update per trajectory.

6.2 Fitted NAC

Fitted natural actor critic (FNAC) is a fitted version of the natural actor critic algorithm [13]. It allows the utilization of general function approximations and efficient data reuse. This combination is especially appealing, because it combines the faster convergence properties of natural gradient algorithms with the efficient data use of regression algorithms.

6.3 Importance Sampling

Melo et. al additionally implemented importance sampling, which increased the efficient use of data even further [13].

6.4 Incremental NAC

“The way we use natural gradients is distinctive in that it is totally incremental: the policy is changed on every time step, yet the gradient computation is never reset as it is in the algorithm of Peters et al. (2005). Alg. 3 is perhaps the most interesting of the three natural-gradient algorithms. It never explicitly stores an estimate of the inverse Fisher information matrix and, as a result, it requires less computation. In empirical experiments using our algorithms (not reported here) we observed that it is easier to nd good parameter settings for Alg. 3 than it is for the other natural-gradient algorithms and, perhaps because of this, it converged more rapidly than the others and than Kondas algorithm.” [3]

6.5 Implicit incremental NAC

6.6 Regularization on NAC

Even if we find the inverse of G, it can be ill defined. An example for this are extremely small eigenvalues which appear due to noise in our 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 will falsify our inverse.
That is why there have been some approaches to introduce a regularization term [21]. Regularizing the matrix inverse can for example be done by a technique called stochastic robust approximation [5], where G^{-1} is replaced by

$$G_{\text{reg}}^{-1} = \left(G^T G + \epsilon I\right)^{-1} G^T \tag{19}$$

where ϵ denotes for a small constant (e.g 0.01).
Another idea is the application of ridge regression [8], which has a build in regularizer. We can calculate $\widetilde{\nabla}_\theta J(\theta)$ by solving the linear equation

$$G(\theta)\widetilde{\nabla}_\theta J(\theta) = \nabla_\theta J(\theta) \tag{20}$$

in the direction of $\widetilde{\nabla}_\theta J(\theta)$.
Witsch et al. use an approach similar to least squares regularization [26].

$$\widetilde{\nabla}_\theta J(\theta) = (F + \lambda I)^{-1} \nabla_\theta J(\theta) \tag{21}$$

If λ is huge, the Fisher matrix only has a small influence on the change in direction. Therefore, we want to scale λ regarding F:

$$\lambda = \frac{\alpha}{\det(F) + 1} \tag{22}$$

with α a small constant, e.g. 0.01.

6.7 POMDPs

There have been some approaches to apply the NAC to POMDPS. A promising approach is the Natural Actor and Belief Critic [10], which learns parameters in statistical dialogue systems. These can be modeled as POMDPs.

6.8 Least Squares

Recursive least squares: [15]

6.9 Truncated Natural Policy Gradient

We use conjugate gradients (CG) to avoid calculating the inverse of the fisher matrix. We apply CG by solving the equation

$$x_k \approx H_k^{-1} g_k \Rightarrow H_k^{-1} x_k \approx g_k \tag{23}$$

If we transform it into an optimization problem for quadratic equations, we have to optimize

$$\min_{x \in \mathbb{R}^n} \frac{1}{2} x^T H x - g^T x \tag{24}$$

7 Discussion

Some sources claim that the eNAC and the NAC-LSTD algorithms use a biased estimate of the natural gradient [24].

References

1. Shun-ichi Amari. Differential geometry of a parametric family of invertible linear systemsriemannian metric, dual affine connections, and divergence. *Mathematical systems theory*, 20(1):53–82, 1987.
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, Mohammad Ghavamzadeh, Mark Lee, and Richard S Sutton. Incremental natural actor-critic algorithms. In *Advances in neural information processing systems*, pages 105–112, 2008.
4. Justin A Boyan. Technical update: Least-squares temporal difference learning. *Machine learning*, 49(2-3):233–246, 2002.
5. Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.
6. 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.
7. 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.
8. Arthur E Hoerl and Robert W Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.
9. Ronald A Howard. Dynamic programming and markov processes. 1960.
10. 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.
11. Byungchan Kim, Jooyoung Park, Shinsuk Park, and Sungchul Kang. Impedance learning for robotic contact tasks using natural actor-critic algorithm. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 40(2):433–443, 2010.
12. Michail G Lagoudakis and Ronald Parr. Least-squares policy iteration. *Journal of machine learning research*, 4(Dec):1107–1149, 2003.
13. 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.
14. Arkadi Nemirovski. Efficient methods in convex programming. 2005.
15. 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.
16. Razvan Pascanu and Yoshua Bengio. Revisiting natural gradient for deep networks. *arXiv preprint arXiv:1301.3584*, 2013.

17. Jan Peters and Stefan Schaal. Applying the episodic natural actor-critic architecture to motor primitive learning. In *ESANN*, pages 295–300, 2007.
18. Jan Peters and Stefan Schaal. Natural actor-critic. *Neurocomputing*, 71(7-9):1180–1190, 2008.
19. Jan Peters, Sethu Vijayakumar, and Stefan Schaal. Natural actor-critic. In *European Conference on Machine Learning*, pages 280–291. Springer, 2005.
20. 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.
21. Jascha Sohl-Dickstein. The natural gradient by analogy to signal whitening, and recipes and tricks for its use. *arXiv preprint arXiv:1205.1828*, 2012.
22. Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.
23. 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.
24. Philip Thomas. Bias in natural actor-critic algorithms. In *International Conference on Machine Learning*, pages 441–448, 2014.
25. Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.
26. 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.