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 Criti · NAC

1 Introduction

Policy gradient methods have dominated the field of reinforcement learning in the last years. These methods build on estimating the policy function by an differentiable function approximation. They optimize an objective function $J(\theta)$ by repeatedly calculating the gradient of the approximated policy $\pi(a|s,\theta)$ w.r.t. the parameters θ and updating the parameters in that direction by stochastic gradient descent.[EDIT: DELETE NAMES OF LIBRARIES?] Python machine learning libraries like Keras, Tensorflow or PyTorch are used for implementing neural networks as differentiable function approximators,

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

which are then trained using the policy gradients of their weights:

$$\Delta_\theta = \alpha \nabla_\theta J(\theta) \tag{1}$$

Simple calculus tells us that the gradient is the steepest direction so eventually this method will converge to a local maximum. So how do we get the gradient of the objective function $J(\theta)$ w.r.t. the parameters θ? Applying some transformations and the likelihood-ratio trick, we obtain the policy gradient theorem [EDIT: REF], which gives us the gradient of the objective function:

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

One of the most basic policy gradient algorithms, the *REINFORCE* algorithm [EDIT: SOURCE], the action-value function Q^{π_θ} is estimated by the Monte-Carlo return:

$$Q^{\pi_\theta} \approx R_\tau = TODO \tag{3}$$

However, one could also use any other admissible estimator. Also estimating the action-value function with a function approximation $Q^{\pi_\theta}(s,a) \approx Q_w(s,a)$ is called an *actor critic method* [EDIT: REF]. One could then apply temporal difference like updates after one or arbitrarily many steps which reduce the variance [EDIT: OF?] because we do not have to roll out the complete episode to update our parameters.
Further, one could introduce a baseline [EDIT: REF]. This does not change the expected value and further reduces our variance. Equation 2 redefines to:

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

where $B(s,a)$ defines the baseline. A good baseline with minimal variance is in many cases the value function of the network. If we take the action-value function and reduce it by the value function, we are left with the advantage function $A(s,a)$. We can immediately estimate the advantage function $A_w(s,a)$, which we will do throughout this paper. Equation 4 changes to:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a|s) A_w(s,a)] \tag{5}$$

Using the stochastic gradient of the objective function to optimize it is very prune to reparameterizations. One could instead use the *natural gradient*, which is described in the next section.

2 Natural Gradient

The natural gradient was first introduced by Amari in 1998 [1]. The main difference between the natural gradient and the ordinary stochastic gradient method, is the direction it points to. Despite many expectations, the ordinary gradient does not always point to the steepest direction, especially not if the parameter space has a Riemann metric structure and is not Euclidean. For example, the parameter space of neural networks has been shown to exhibit a

Riemannian character.
In comparison to vanilla gradient methods, the natural gradient finds the steepest direction for Riemannian parameter spaces. It utilizes the $n \times n$ matrix $G = (g_{ij})$, called the Riemannian metric tensor, which in general depend on θ. The natural gradient is defined as

$$\widetilde{\nabla}_\theta J(\theta) = G^{-1} \nabla_\theta J(\theta) \tag{6}$$

where $\widetilde{\nabla}_\theta$ is the natural gradient w.r.t the parameters θ. Learning should be carried out with an gradient descent like update rule:

$$\theta_{t+1} = \theta_{t+1} + \alpha \widetilde{\nabla}_\theta J(\theta) \tag{7}$$

In the special case that the parameter space is Euclidean and the coordinate system is orthonormal, the conventional gradient equals the natural gradient:

$$\widetilde{\nabla}_\theta J(\theta) = \nabla_\theta \tag{8}$$

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

$$F_\theta = \mathbb{E}_{s \sim \rho^\pi, a \sim \pi_\theta} \left[\nabla_\theta log \pi_\theta(a|s)^T \nabla_\theta log \pi_\theta(a|s) \right] \tag{9}$$

For deterministic policies:

$$M_\mu(\theta) = E_{s \sim \rho^\mu} [\nabla_\theta \mu_\theta(s) \nabla_\theta \mu_\theta(s)^T w] \tag{10}$$

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

$$\nabla_\theta J(\mu_\theta) = E_{s \sim \rho^\mu} [\nabla_\theta \mu_\theta(s) \nabla_\theta \mu_\theta(s)^T w] \tag{11}$$

so steepest ascent direction reduces to

$$M_\mu(\theta)^{-1} \nabla_\theta J_\beta(\mu_\theta) = w \tag{12}$$

Natural gradient algorithms have been applied successfully in various fields such as road traffic optimization [18], robotic control tasks [9], motor primitive learning [14] and locomotion of a two-linked robot arm [12]. Using the natural gradient has several advantages and properties, which are listed in the next section.

3 Properties of the natural gradient

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

3.1 Online Learning

The natural gradient can be used online. This means that we can learn from incomplete sequences and by this reduce the variance [EDIT:OF?]. 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 [13, 15].

3.2 1st order method

The natural gradient is a first order method, but implements many second order advantages [13]. 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.

3.3 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 [13]. This is comparable to signal noise whitening [19].

3.4 Faster convergence

In many cases natural gradient algorithms converge faster than vanilla gradient algorithms [19, 1]. 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 [19].

3.5 Better convergence properties

Using the natural gradient instead of the vanilla gradient avoids getting stuck in local optima during optimization [1].

3.6 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 [11].

3.7 More properties

Look at Berkley slides

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

4 Natural Actor Critic

In this section we present an episodic algorithm called *Natural Actor Critic* (NAC) [17] using the methods mentioned above. The pseudo-code for NAC can be found in algorithm 1. We show several possible extensions to the algorithm in the next section. [EDIT: DRAWBACKS? WHY EXTENSIONS?]

Algorithm 1 Episodic Natural Actor Critic Algorithm (eNAC)

```
Require: Parameterized policy $\pi(a|s,\theta)$ with initial parameters $\theta = \theta_0$
Require: It's derivative $\nabla_\theta \log \pi(a|s,\theta)$
  for $u = 1, 2, 3, \dots$ do
    for $e = 1, 2, 3, \dots$ do
      Execute roll-out: Draw initial state $s_0 \sim p(s_0)$
      for $t = 1, 2, 3, \dots, N$ do
        Draw action $a_t \sim \pi(a_t|s_t,\theta_t)$, observe next state $s_{t+1} \sim p(s_{t+1}|s_t,a_t)$, and reward $r_t = r(s_t,a_t)$.
      end for
    end for
    Critic Evaluation (Episodic): Determine the advantage function by using compatible function approximation: $A(s,a) \approx \hat{A}(s,a|w)$
    Update: Determine basis functions: $\phi_t = \left[\sum_{t=0}^{T} \gamma^t \nabla_\theta \log \pi(a_t|s_t,\theta)^T, 1\right]^T$: reward statistics: $R_t = \sum_{t=0}^{T} \gamma^t r_t$
    Actor Update: When the natural gradient is converged $((w_{t+1}, w_t) \leq \epsilon)$, update the policy parameters: $\theta_{t+1} = \theta_t + \alpha w_{t+1}$
  end for
```

5 Extensions

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

5.1 Episodic NAC

5.2 Fitted NAC

Fitted natural actor critic (FNAC) is a fitted version of the natural actor critic algorithm [10]. 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.

5.3 Importance Sampling

Melo et. al additionally implemented importance sampling, which increased the efficient use of data even further [10].

5.4 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." [2]

5.5 Implicit incremental NAC

5.6 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 [19]. Regularizing the matrix inverse can for example be done by a technique called stochastic robust approximation [4], where G^{-1} is replaced by

$$G_{\text{reg}}^{-1} = \left(G^T G + \epsilon I\right)^{-1} G^T \tag{13}$$

where ϵ denotes for a small constant (e.g 0.01).

Another idea is the application of ridge regression [7], which has a build in regularizer. We can calculate $\widetilde{\nabla}_\theta J(\theta)$ by solving the linear equation

$$G(\theta)\widetilde{\nabla}_\theta J(\theta) = \nabla_\theta J(\theta) \tag{14}$$

in the direction of $\widetilde{\nabla}_\theta J(\theta)$.

Witsch et al. use an approach similar to least squares regularization [22].

$$\widetilde{\nabla}_\theta J(\theta) = (F + \lambda I)^{-1} \nabla_\theta J(\theta) \tag{15}$$

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:

$$\lambda = \frac{\alpha}{\det(F) + 1} \tag{16}$$

with α a small constant, e.g. 0.01.

5.7 POMDPs

There have been some approaches to apply the NAC to POMDPS. A promising approach is the Natural Actor and Belief Critic [8], which learns parameters in statistical dialogue systems. These can be modeled as POMDPs.

5.8 Least Squares

Recursive least squares: [12]

5.9 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

$$x_k \approx H_k^{-1} g_k \Rightarrow H_k^{-1} x_k \approx g_k \tag{17}$$

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

$$\min_{x \in \mathbb{R}^n} \frac{1}{2} x^T H x - g^T x \tag{18}$$

6 Discussion

Some sources claim that the eNAC and the NAC-LSTD algorithms use a biased estimate of the natural gradient [21].

7 Paper

Natural Actor Critic:

- Main Version [17].
- 2nd Version: Natural Actor-Critic in Neurocomputing [15].
- 3rd version: RL of motor skills with policy gradients in NN [16].

Must read paper to understand basics by Jan:

- Policy Evaluation with TD [5].

Recommended by Jan:

- Incremental NAC algorithms [2].
- Jan said that a paper form C. Dann is very important. Did he mean Policy Evaluation with TD by Dann or did he mean a second paper?

Research:

- Comparison of four natural gradient algorithms (co-author Sutton) [3].

8 Meetings & Notes

Meetings:

- 12.12.18: Notes from Jan can be found in “.\Notes Jan 12.12.18”

9 Ideas

9.1 Fitted NAC

$$\pi(a|s) = p(a|s,\Theta) = \mathcal{N}(a|\mu = NN_\Theta(s), \sigma) \tag{19}$$

$$f(s,a) = \log p(a|s,\Theta)^T w \tag{20}$$

$$f_V(s) = NN_V(s) \tag{21}$$

$$\text{"fitted"} \tag{22}$$

$$\min_{V_t, W_t} (r(s,a) + \gamma f_{V_{t+1}}(s') - f_{V_t}(s) + f_{W_t}(s,a))^2 \tag{23}$$

9.2 Policy Evaluation with Temporal Difference

[5]

9.3 Ideas of TRPO

9.4 Other Extentions

– stochastic
– minibatches
– importance sampling:

$$V_\Theta J = \sum \mu(s)\pi'(a|s)\nabla \log \pi'(a|s)Q^{\pi'}(a|s) \tag{24}$$
$$\approx \frac{1}{N}\sum \nabla \log \pi'(a|s)Q(s,a) = g(\Theta) \tag{25}$$

10 Episodic NAC

Important to understand beforehand: In episodic NAC, our system of equations has one equation per trajectory and not one equation per action as in the normal NAC algorithm.

First we start by adding together the advantage function across an episode e where we made N steps.

$$A(s,a) = r(s,a) + \gamma V(s') - V(s) \tag{26}$$
$$\gamma A(s',a') = \gamma r(s',a') + \gamma^2 V(s'') - \gamma V(s') \tag{27}$$
$$A(s,a) + \gamma A(s',a') = r(s,a) + \gamma r(s',a') + \gamma^2 V(s'') - V(s) \tag{28}$$
$$\sum_{i=0}^{N} \gamma^i A(s_i,a_i) = \sum_{i=0}^{N} \gamma^i r(s_i,a_i) + \gamma^N V(S_{N+1}) - V(S_0) \tag{29}$$

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 ($\gamma^N \to 0$). Additionally, if we assume that we always start in the same start S_0, we can write $V(S_0)$ as our cost function J because it will exactly sum up the expected Reward/cost of our problem.

$$\Rightarrow \sum_{i=0}^{N} \gamma^i A(s_i,a_i) = \sum_{i=0}^{N} \gamma^i r(s_i,a_i) - J \tag{30}$$

Now we can plug in the parametrisized gradient descent for the advantage function. That this works and is indeed the same has been proven by reference. Additionally we bring the cost J to the other side of the equation.

$$\Rightarrow \sum_{i=0}^{N} \gamma^i \nabla_\Theta \left[\log \pi(a_i|s_i)^T\right] \cdot w + 1 \cdot J = \sum_{i=0}^{N} \gamma^i r(s_i,a_i) \tag{31}$$

Let's do some rewriting. We define the following two terms:

$$\Phi_e = \left[\sum_{i=0}^{N} \gamma^i \nabla_{\Theta} \left[\log \pi(a_i|s_i)^T \right], 1 \right] \tag{32}$$

$$R_e = \sum_{i=0}^{N} \gamma^i r(s_i, a_i) \tag{33}$$

This let's us rewrite equation 31 as:

$$\Phi_e \cdot \begin{bmatrix} w \\ J \end{bmatrix} = R_e \tag{34}$$

An easy way to solve this system of equations is by taking the pseudo inverse of Φ_e.

$$\begin{bmatrix} w \\ J \end{bmatrix} = (\Phi_e^T \Phi_e)^{-1} \Phi_e^T R_e \tag{35}$$

References

1. 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.
2. 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.
3. Shalabh Bhatnagar, Richard S Sutton, Mohammad Ghavamzadeh, and Mark Lee. Natural actor–critic algorithms. *Automatica*, 45(11):2471–2482, 2009.
4. Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.
5. Christoph Dann, Gerhard Neumann, and Jan Peters. Policy evaluation with temporal differences: A survey and comparison. *The Journal of Machine Learning Research*, 15(1):809–883, 2014.
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. Arthur E Hoerl and Robert W Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.
8. 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.

9. 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.
10. 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.
11. Arkadi Nemirovski. Efficient methods in convex programming. 2005.
12. 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.
13. Razvan Pascanu and Yoshua Bengio. Revisiting natural gradient for deep networks. *arXiv preprint arXiv:1301.3584*, 2013.
14. Jan Peters and Stefan Schaal. Applying the episodic natural actor-critic architecture to motor primitive learning. In *ESANN*, pages 295–300, 2007.
15. Jan Peters and Stefan Schaal. Natural actor-critic. *Neurocomputing*, 71(7-9):1180–1190, 2008.
16. Jan Peters and Stefan Schaal. Reinforcement learning of motor skills with policy gradients. *Neural networks*, 21(4):682–697, 2008.
17. Jan Peters, Sethu Vijayakumar, and Stefan Schaal. Natural actor-critic. In *European Conference on Machine Learning*, pages 280–291. Springer, 2005.
18. 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.
19. Jascha Sohl-Dickstein. The natural gradient by analogy to signal whitening, and recipes and tricks for its use. *arXiv preprint arXiv:1205.1828*, 2012.
20. 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.
21. Philip Thomas. Bias in natural actor-critic algorithms. In *International Conference on Machine Learning*, pages 441–448, 2014.
22. 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.