Fitted Natural Actor-Critic: A New Algorithm for Continuous State-Action MDPs*

Francisco S. Melo[1] and Manuel Lopes[2]

[1] Carnegie Mellon University
Pittsburgh, USA
fmelo@isr.ist.utl.pt
[2] Institute for Systems and Robotics
Lisboa, Portugal
macl@isr.ist.utl.pt

Abstract. In this paper we address reinforcement learning problems with continuous state-action spaces. We propose a new algorithm, *fitted natural actor-critic* (FNAC), that extends the work in [1] to allow for general function approximation and data reuse. We combine the natural actor-critic architecture [1] with a variant of fitted value iteration using importance sampling. The method thus obtained combines the appealing features of both approaches while overcoming their main weaknesses: the use of a gradient-based actor readily overcomes the difficulties found in regression methods with policy optimization in continuous action-spaces; in turn, the use of a regression-based critic allows for efficient use of data and avoids convergence problems that TD-based critics often exhibit. We establish the convergence of our algorithm and illustrate its application in a simple continuous space, continuous action problem.

1 Introduction

In theory, reinforcement learning (RL) can be applied to address any optimal control task, yielding optimal solutions while requiring very little *a priori* information on the system itself. Existing RL methods are able to provide optimal solutions for many real-world control problems featuring discrete state and action spaces and exhibit widely studied convergence properties [2]. However, most such methods do not scale well in problems with large state and/or action spaces.

Many RL works addressing problems with infinite state-spaces combine function approximations with learning methods. Encouraging results were reported, perhaps the most spectacular of which by Tesauro and his learning Gammon player [3]. However, as seen in [4, 5], DP/TD-based methods exhibit unsound

* Work partially supported by the Information and Communications Technologies Institute, the Portuguese Fundação para a Ciência e a Tecnologia, under the Carnegie Mellon-Portugal Program, the Programa Operacional Sociedade de Conhecimento (POS_C) that includes FEDER funds and the project PTDC/EEA-ACR/70174/2006, and by the EU Project (IST-004370) RobotCub.

W. Daelemans et al. (Eds.): ECML PKDD 2008, Part II, LNAI 5212, pp. 66–81, 2008.
© Springer-Verlag Berlin Heidelberg 2008

convergence behavior when combined with general function approximation. Convergence of methods such as Q-learning with general function approximation thus remains an important open issue [6].

If problems with infinite state-spaces pose important challenges when developing efficient RL algorithms, the simultaneous consideration of infinite action-spaces adds significant difficulties. Few RL methods to this day address problems featuring continuous state and action spaces. A fundamental issue in this class of problems is *policy optimization*: many RL methods rely on explicit maximization of a utility function to achieve policy optimization. If the number of available actions is *infinite*, this maximization is generally hard to achieve, especially if we consider that it is not *local* but *global* maximization. This significant difficulty affects many methods with otherwise sound performance guarantees, rendering such performance guarantees unusable [7].

When addressing problems with large/infinite state and/or action spaces, two major approaches have been considered in the RL literature. *Regression-based methods* use sample data collected from the system to estimate some target utility function using regression techniques. This class of methods is particularly suited to address problems with infinite state-spaces, although more general applications have been proposed in the literature [7]. Such algorithms can take advantage of the numerous regression methods available from the machine learning literature while exhibiting solid convergence properties [7, 8] and have been successfully applied in many different problems [9, 10, 11, 12].

Gradient-based methods, on the other hand, are naturally suited to address problems with infinite action-spaces. Such methods consider a parameterized policy and estimate the gradient of the performance with respect to the policy parameters. The parameters are then updated in the direction of this estimated gradient. By construction, gradient-based methods implement an incremental policy optimization and thus avoid the need for explicit maximization; it is no surprise that many RL works addressing problems with continuous action spaces thus rely on a gradient-based architecture [1, 13, 14].

However, gradient-based methods are line-search methods and, therefore, convergence is guaranteed only to local minima. Moreover, "pure" gradient methods usually exhibit large variance and, as argued in [15], make poor use of data. *Actor-critic architectures* [16] provide a suitable extension to pure gradient methods. They have been extensively analyzed in several works (*e.g.*, [1, 15, 17, 18]) and found to exhibit several advantages over pure gradient methods (in particular, in terms of variance and data-usage).

1.1 Contributions and Structure of the Paper

In this paper, we combine the appealing properties of actor-critic methods with those of regression-based methods in a new method dubbed as *fitted natural actor-critic* (FNAC). FNAC extends the natural actor-critic (NAC) architecture [1], allowing general function approximation and data reuse. In particular, we modify the TD-based critic in the NAC and implement a variant of fitted value iteration using importance sampling.

FNAC thus combines in a single algorithm the potentially faster convergence of *natural gradients* [19] and the sound convergence properties and efficient use of data of regression algorithms [7]. We also make use of an *importance sampling* strategy that allows the reuse of data, making our algorithm very efficient in terms of data usage and allowing the analysis of convergence of the algorithm.[1]

To this respect, it is also worth mentioning that, in many practical problems, collecting data is costly and time-consuming. In these situations, the efficient use of data in FNAC is a significant advantage over other existing approaches. Finally, it is also important to emphasize that the gradient-based policy updates readily overcome the most obvious difficulties of stand-alone regression-methods with respect to policy optimization. Summarizing, FNAC allows for *general function approximation* in the critic component, while *being able to use all sampled data* in all iterations of the algorithm (unlike most previous methods).

The paper is organized as follows. Section 2 reviews some background material on MDPs and policy gradient methods. Section 3 introduces the fitted natural actor-critic algorithm and its main properties. We evaluate its performance in the continuous mountain-car problem in Section 4 and conclude in Section 5 with some final remarks and directions for future work.

2 Background

In this section we review some background material that will be of use in the remainder of the paper. In particular, we briefly review the MDP framework [20], the policy gradient theorem and its application in approximate settings [21] and the use of natural gradients in actor-critic methods [1, 19].

2.1 Markov Decision Problems

Given two compact sets $\mathcal{X} \subset \mathbb{R}^p$ and $\mathcal{A} \subset \mathbb{R}^q$, let $\{X_t\}$ be an $\mathcal{X}$-valued controlled Markov chain, with control parameter taking values in $\mathcal{A}$. The transition probabilities for the chain are given by the kernel

$$\mathbb{P}\left[X_{t+1} \in U_X \mid X_t = x, A_t = a\right] = \mathsf{P}_a(x, U_X),$$

for any measurable set $U_X \subset \mathcal{X}$. The $\mathcal{A}$-valued process $\{A_t\}$ represents the control process: A_t is the control action at time instant t. A decision-maker must determine the control process $\{A_t\}$ so as to maximize the functional

$$V(\{A_t\}, x) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(X_t, A_t) \mid X_0 = x\right],$$

where $0 \leq \gamma < 1$ is a discount-factor and $R(x, a)$ represents a random "reward" received for taking action $a \in \mathcal{A}$ in state $x \in \mathcal{X}$. To play it safe, we assume

[1] As remarked in [18], such analysis is not immediate in the original NAC algorithm, since the data used in NAC is generated online from the current policy estimate in the algorithm.

throughout this paper that there is a deterministic continuous function r defined on $\mathcal{X} \times \mathcal{A} \times \mathcal{X}$ assigning a reward $r(x, a, y)$ every time a transition from x to y occurs after taking action a and that

$$\mathbb{E}\left[R(x,a)\right] = \int_{\mathcal{X}} r(x,a,y)\mathsf{P}_a(x,dy).$$

This simplifies the notation without introducing a great loss in generality. We further assume that there is a constant $\mathcal{R} \in \mathbb{R}$ such that $|r(x, a, y)| < \mathcal{R}$ for all $x, y \in \mathcal{X}$ and all $a \in \mathcal{A}$. We refer to the 5-tuple $(\mathcal{X}, \mathcal{A}, \mathsf{P}, r, \gamma)$ as a *Markov decision problem* (MDP).

Given an MDP $\mathcal{M} = (\mathcal{X}, \mathcal{A}, \mathsf{P}, r, \gamma)$, the *optimal value function* V^* is defined for each state $x \in \mathcal{X}$ as

$$V^*(x) = \max_{\{A_t\}} \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^t R(X_t, A_t) \mid X_0 = x\right]$$

and verifies

$$V^*(x) = \max_{a \in \mathcal{A}} \int_{\mathcal{X}} \left[r(x,a,y) + \gamma V^*(y)\right]\mathsf{P}_a(x,dy), \tag{1}$$

which is a form of the Bellman optimality equation.[2] The optimal Q-values $Q^*(x, a)$ are defined for each state-action pair $(x, a) \in \mathcal{X} \times \mathcal{A}$ as

$$Q^*(x,a) = \int_{\mathcal{X}} \left[r(x,a,y) + \gamma V^*(y)\right]\mathsf{P}_a(x,dy).$$

From Q^* we can define the mapping $\pi^*(x) = \arg\max_a Q^*(x, a)$ for all $x \in \mathcal{X}$. The control process defined by $A_t = \pi^*(X_t)$ is optimal in the sense that the corresponding value function equals V^*. The mapping π^* thus defined is an *optimal policy* for the MDP $\mathcal{M}$.

More generally, a *policy* is a (time-dependent) mapping π_t defined over $\mathcal{X} \times \mathcal{A}$ that generates a control process $\{A_t\}$ verifying

$$\mathbb{P}\left[A_t \in U_A \mid X_t = x\right] = \int_{U_A} \pi_t(x,a)da, \qquad \forall t,$$

where $U_A \subset \mathcal{A}$ is any measurable set. We write $V^{\pi_t}(x)$ instead of $V(\{A_t\}, x)$ if the control process $\{A_t\}$ is generated by a policy π_t. A *stationary policy* is a policy π that does not depend on t.

2.2 The Policy Gradient Theorem

Let π^θ be a stationary policy parameterized by some finite-dimensional vector $\theta \in \mathbb{R}^M$. Assume, in particular, that π is continuously differentiable with respect

[2] Notice that the maximum in (1) is well defined due to our assumption of compact $\mathcal{A}$ and continuous r.

to (w.r.t.) θ. We henceforth write V^θ instead of V^{π^θ} to denote the corresponding value function. Also, given some probability measure μ_0 over $\mathcal{X}$, we define

$$\rho(\theta) = (\mu_0 V^\theta) = \int_{\mathcal{X}} V^\theta(x)\mu_0(dx).$$

We abusively write $\rho(\theta)$ instead of $\rho(\pi^\theta)$ to simplify the notation. The function $\rho(\theta)$ can be seen as the total discounted reward that an agent expects to receive when following the policy π^θ and the initial state is distributed according to μ_0.

We wish to compute the parameter vector θ^* such that the corresponding policy π^{θ^*} maximizes the expected income for the agent in the sense of ρ. In other words, we wish to compute $\theta^* = \arg\max_\theta \rho(\theta)$. If ρ is differentiable w.r.t. θ, this can be achieved by updating θ according to

$$\theta_{t+1} = \theta_t + \alpha_t \nabla_\theta \rho(\theta_t),$$

where $\{\alpha_t\}$ is a step-size sequence and ∇_θ denotes the gradient w.r.t. θ. We can now introduce the following result from [21, 22].

Theorem 1. *Given an MDP $\mathcal{M} = (\mathcal{X}, \mathcal{A}, \mathsf{P}, r, \gamma)$, it holds for every $x \in \mathcal{X}$ that*

$$\nabla_\theta V^\theta(x) = \int_{\mathcal{X}\times\mathcal{A}} \nabla_\theta \pi^\theta(y,a) Q^\theta(y,a) da\ \hat{\mathsf{K}}_\gamma^\theta(x, dy),$$

where $\hat{\mathsf{K}}_\gamma^\theta$ is the un-normalized γ-resolvent associated with the Markov chain induced by π^θ.[3]

The fact that $\rho(\theta) = (\mu_0 V^\theta)$ immediately implies that

$$\nabla_\theta \rho(\theta) = \int_{\mathcal{X}} \nabla_\theta V^\theta(x)\mu_0(dx).$$

For simplicity of notation, we henceforth denote by μ_γ^θ the measure over $\mathcal{X}$ defined by

$$\mu_\gamma^\theta(U_X) = \int_{\mathcal{X}} \left(\int_{U_X} \hat{\mathsf{K}}_\gamma^\theta(x, dy)\right) \mu_0(dx).$$

2.3 Policy Gradient with Function Approximation

From Theorem 1 it is evident that, in order to compute the gradient $\nabla\rho$, the function Q^θ needs to be computed. However, when addressing MDPs with infinite state and/or action spaces (as is the case in this paper), some form of function approximation is needed in order to compute Q^θ.

[3] The γ-resolvent [23] associated with a Markov chain $(\mathcal{X}, \mathsf{P})$ is the transition kernel K_γ defined as $\mathsf{K}_\gamma(x, U) = (1-\gamma)\sum_{t=0}^\infty \gamma^t \mathsf{P}^t(x, U)$ and the un-normalized γ-resolvent is simply $\hat{\mathsf{K}}_\gamma(x, U) = \sum_{t=0}^\infty \gamma^t \mathsf{P}^t(x, U)$.

Let $\{\phi_i, i = 1, \ldots, M\}$ be a set of M linearly independent functions and $\mathcal{L}(\phi)$ its linear span. Let $\hat{Q}^\theta$ be the best approximation of Q^θ in $\mathcal{L}(\phi)$, taken as the orthogonal projection of Q^θ on $\mathcal{L}(\phi)$ w.r.t. the inner product

$$\langle f, g \rangle = \int_{\mathcal{X} \times \mathcal{A}} f(x, a) \cdot g(x, a) \pi^\theta(x, a) da \ \mu_\gamma^\theta(dx).$$

As any function in $\mathcal{L}(\phi)$, $\hat{Q}^\theta$ can be written as

$$\hat{Q}^\theta(x, a) = \sum_i \phi_i(x, a) w_i = \phi^\top(x, a) w.$$

This leads to the following result from [21].

Theorem 2. *Given an MDP $\mathcal{M} = (\mathcal{X}, \mathcal{A}, \mathsf{P}, r, \gamma)$ and a set of basis functions $\{\phi_i, i = 1, \ldots, M\}$ as defined above, if*

$$\nabla_w \hat{Q}^\theta(x, a) = \nabla_\theta \log(\pi^\theta(x, a)) \tag{2}$$

then

$$\nabla_\theta \rho(\theta) = \int_{\mathcal{X} \times \mathcal{A}} \nabla_\theta \pi^\theta(x, a) \hat{Q}^\theta(x, a) da \ \mu_\gamma^\theta(dx).$$

Notice that, in the gradient expression in Theorems 1 and 2, we can add an arbitrary function $b(x)$ to Q^θ and $\hat{Q}^\theta$. This is clear from noting that

$$\int_{\mathcal{A}} \nabla_\theta \pi^\theta(x, a) b(x) da = 0.$$

Such functions are known as *baseline functions* and, as recently shown in [18], if b is chosen so as to minimize the mean-squared error between $\hat{Q}^\theta$ and Q^θ, the optimal choice of baseline function is $b(x) = V^\theta(x)$. Recalling that the *advantage function* [24] associated with a policy π is defined as $A^\pi(x, a) = Q^\pi(x, a) - V^\pi(x)$, we get

$$\nabla_\theta \rho(\theta) = \int_{\mathcal{X} \times \mathcal{A}} \nabla_\theta \pi^\theta(x, a) \hat{A}^\theta(x, a) da \ \mu_\gamma^\theta(dx), \tag{3}$$

where $\hat{A}^\theta(x, a)$ denotes the orthogonal projection of the advantage function associated with π^θ, A^θ, into $\mathcal{L}(\phi)$. Finally, recalling that $\hat{A}^\theta(x, a) = \phi^\top(x, a) w$, we can compactly write (3) as $\nabla_\theta \rho(\theta) = \mathbf{G}(\theta) w$, with $\mathbf{G}$ the *all-action matrix* [1].

$$\mathbf{G}(\theta) = \int_{\mathcal{X} \times \mathcal{A}} \nabla_\theta \pi^\theta(x, a) \phi^\top(x, a) da \ \mu_\gamma^\theta(dx). \tag{4}$$

2.4 Natural Gradient

Given a general manifold M parameterized by a finite-dimensional vector $\theta \in \mathbb{R}^M$ and a real function F defined over this manifold, the gradient $\nabla_\theta F$ seldom corresponds to the actual steepest descent direction, as it fails to take into account the geometry of the manifold [25]. However, in many practical situations, it is possible to impose a particular structure on the manifold (namely, a Riemannian metric) and compute the steepest descent direction taking into account the geometry of the manifold (in terms of the Riemannian metric). This "natural gradient" is invariant to changes in parameterization of the manifold and can potentially overcome the so-called plateau phenomenon [25].

As seen in [19], the parameterized policy space can be seen as a manifold that can be endowed with an adequate Riemannian metric. One possible metric relies on the *Fisher information matrix*, and is defined by the following matrix [19]

$$\mathbf{F}(\theta) = \int_{\mathcal{X}\times\mathcal{A}} \nabla_\theta \log(\pi^\theta(x,a)) \nabla_\theta \log(\pi^\theta(x,a))^\top \pi^\theta(x,a) da\ \mu_\gamma^\theta(dx).$$

The natural gradient is given, in this case, by $\tilde{\nabla}_\theta \rho(\theta) = \mathbf{F}^{-1}(\theta)\mathbf{G}(\theta)w$. However, as shown in [1], multiplying and dividing the integrand in (4) by $\pi^\theta(x,a)$, we get $\mathbf{G}(\theta) = \mathbf{F}(\theta)$, and the natural gradient comes, simply, $\tilde{\nabla}_\theta \rho(\theta) = w$.[4]

3 Fitted Natural Actor-Critic

We now use the ideas from the previous section to derive a new algorithm, *fitted natural actor-critic* (FNAC). This algorithm takes advantage of several appealing properties of fitting methods (namely, the solid convergence guarantees and the effective use of data) while overcoming some of the limitations of this class of methods in problems with continuous action spaces.

3.1 The FNAC Architecture

We start by briefly going through the FNAC architecture, illustrated in Figure 1.

The algorithm uses a set $\mathcal{D}$ of samples obtained from the environment, each consisting of a tuple (x_t, a_t, r_t, y_t), where y_t is a sample state distributed according to the measure $\mathsf{P}_{a_t}(x_t, \cdot)$ and $r_t = r(x_t, a_t, y_t)$. For the purposes of the algorithm, it is not important how the samples in D are collected. In particular, they can all be collected before the algorithm is run or they can be collected incrementally, as more iterations of the algorithm are performed. Nevertheless, it is important to remark that, for the purposes of our critic, enough data-samples need to be collected to avoid conditioning problems in the regression algorithms.

At each iteration of the FNAC algorithm, the data in $\mathcal{D}$ is processed by the *critic* component of the algorithm. This component, as detailed below, uses a generic regression algorithm to compute an approximation $\hat{V}^\theta$ of the value function associated with the current policy, π^θ. This approximation is then used to estimate an approximation of the advantage function, $\hat{A}^\theta$, using a linear function approximation with compatible basis functions. Finally, the evaluation

[4] Peters et al. [1] actually showed that $\mathbf{F}(\theta)$ is the Fisher information matrix for the probability distribution over possible trajectories associated with a given policy.

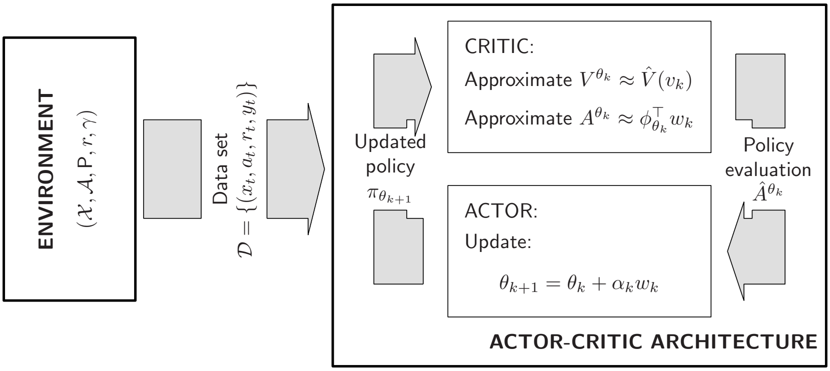
ENVIRONMENT
$(\mathcal{X}, \mathcal{A}, \mathsf{P}, r, \gamma)$
Data set
$\mathcal{D} = \{(x_t, a_t, r_t, y_t)\}$
Updated
policy
$\pi_{\theta_{k+1}}$
CRITIC:
Approximate $V^{\theta_k} \approx \hat{V}(v_k)$
Approximate $A^{\theta_k} \approx \phi_{\theta_k}^\top w_k$
ACTOR:
Update:
$\theta_{k+1} = \theta_k + \alpha_k w_k$
Policy
evaluation
$\hat{A}^{\theta_k}$
ACTOR-CRITIC ARCHITECTURE

Fig. 1. Detailed FNAC architecture

performed by the critic (*i.e.*, the approximation $\hat{A}^\theta$ computed from the data) is used by the actor to update the policy π^θ using a standard policy gradient update.

In the remaining of this section, we detail our FNAC algorithm.

3.2 The Actor

The actor component of the FNAC algorithm implements a policy gradient update. As seen in Section 2, this update relies on the *natural gradient*. The fact that the natural gradient manages to take into account the geometry of the policy space may potentially bring significant advantages in terms of performance of the algorithm (namely, in terms of rate of convergence and ability to overcome the plateau phenomenon) [25]. Therefore, given the parameterized policy at iteration k, π_{θ_k}, the actor component will update the parameter vector as

$$\theta_{k+1} = \theta_k + \alpha_k \tilde{\nabla}_\theta \rho(\theta_k).$$

As seen in Subsection 2.4, the natural gradient is given by $\tilde{\nabla}\rho(\theta_k) = w_k$, where w_k is the linear coefficient vector corresponding to the orthogonal projection of the advantage function A^{θ_k} in the linear space spanned by the *compatible basis functions*, obtained from 2:

$$\phi_i(x, a) = \frac{\partial \log(\pi_{\theta_k})}{\partial \theta_k(i)}(x, a).$$

Therefore, provided that the critic component yields such an approximation of the advantage function, the update rule for the actor is, simply, $\theta_{k+1} = \theta_k + \alpha_t w_k$.

3.3 The Critic

The critic of the FNAC algorithm is the element that distinguishes our algorithm from other gradient-based approaches, such as [1, 18]. Although we discuss these differences in detail in the next subsection, it is still worth to briefly outline the fundamental differences. The algorithm in [18] can be seen as an online version of the algorithm in [1]. Our algorithm is closest to that in [1] (it is, by construction, a batch algorithm, although it can straightforwardly be converted in an online algorithm). However, FNAC allows for more efficient use of data than any of the two aforementioned algorithms and is designed so as to accommodate general regression methods (and, thus, general approximations).

Let $\mathcal{D} = \{(x_t, a_t, r_t, x_{t+1}), t = 1, \ldots, n\}$ be a sequence of sample transitions obtained from the MDP when following some policy π_0. As seen before, the value function associated with a general policy π verifies

$$V^\pi(x) = \int_{\mathcal{X}\times\mathcal{A}} \big(r(x, a, y) + \gamma V^\pi(y)\big)\mathsf{P}_a(x, dy)\pi(x, a)da$$

or, equivalently,

$$\int_{\mathcal{X}\times\mathcal{A}} \big(r(x, a, y) + \gamma V^\pi(y)\big)\mathsf{P}_a(x, dy)\pi(x, a)da - V^\pi(x) = 0.$$

This can be written as

$$\int_{\mathcal{X}\times\mathcal{A}} \big(r(x, a, y) + \gamma V^\pi(y) - V^\pi(x)\big)\mathsf{P}_a(x, dy)\pi(x, a)da = 0.$$

We want to approximate V^π by a general parameterized family of functions $\mathcal{V} = \big\{V_v(x) \mid v \in \mathbb{R}^N\big\}$. In other words, we want to compute v^* such that $V^\pi(x) \approx V_{v^*}(x)$. For the policy π_0 used to generate the dataset $\mathcal{D}$, we can use the data in $\mathcal{D}$ and solve the following regression problem:

$$v^* = \arg\min_v \sum_t \frac{1}{\hat{\mu}(x_t)}\big(r_t + \gamma V_v(y_t) - V_v(x_t)\big)^2, \tag{5}$$

where $\hat{\mu}(x_t)$ is the empirical distribution of state x obtained from the dataset $\mathcal{D}$.[5] We remark that the function V_{v^*} thus obtained is the one minimizing the *empirical Bellman residual.* However, in order to adequately perform the above minimization, double sampling is necessary, as pointed out in [4]. In systems where double sampling is not possible, a correction term can be included in the regression to avoid negative correlation effects [26], rendering the regression problem equivalent to the solution of the following fitted-VI iteration:

$$V_{k+1} = \min_V \sum_t \frac{1}{\hat{\mu}(x_t)}\big(r_t + \gamma V_k(y_t) - V(x_t)\big)^2.$$

[5] The inclusion of the term $\hat{\mu}(x_t)$ merely ensures that regions of the state-space that happen to appear more often *in the dataset* do not have “extra weight” in the regression. In fact, since the data in $\mathcal{D}$ can be obtained by any arbitrary sampling process, its distribution will generally be distinct from that induced by the obtained policy. The “normalization” w.r.t. $\hat{\mu}(x_t)$ minimizes any bias that the sampling process may introduce in the regression. Alternative regularizations are possible, however.

Nevertheless, the computation of V^π as described above is possible because the data is *distributed according to the policy* π_0.

Suppose now that we want to approximate the value function associated with some other policy $\pi^\theta \neq \pi_0$. The value function for this policy verifies

$$V^\theta(x) = \int_{\mathcal{X}\times\mathcal{A}} \big(r(x,a,y) + \gamma V^\theta(y)\big)\mathsf{P}_a(x,dy)\pi^\theta(x,a)da.$$

If we want to use the same data to compute V^θ, some modification is needed since the data in $\mathcal{D}$ is not distributed according to π^θ. If the policy π^θ is known, the above expression can be modified to yield

$$\begin{aligned}\int_{\mathcal{X}\times\mathcal{A}} \big(r(x,a,y) + \gamma V^\pi(y) - V^\pi(x)\big)\mathsf{P}_a(x,dy)\cdot&\\ \cdot\frac{\pi^\theta(x,a)}{\pi_0(x,a)}\pi_0(x,a)da = 0.&\end{aligned} \tag{6}$$

This means that we should be able to reuse the data in $\mathcal{D}$ to solve the following regression problem, similar to that in (5):

$$v^* = \arg\min_v \sum_t \frac{1}{\hat{\mu}(x_t)}\frac{\pi^\theta(x_t,a_t)}{\pi_0(x_t,a_t)}\big(r_t + \gamma V_v(y_t) - V_v(x_t)\big)^2.$$

Notice that the above regression makes use of *importance sampling*, by including the term $\frac{\pi^\theta(x,a)}{\pi_0(x,a)}$. Notice also that this importance-sampling term is well-defined for all samples (x_t, a_t, r_t, y_t), since $\pi_0(x_t,a_t) > 0$ necessarily holds. Notice also that, as before, the term $\hat{\mu}(x_t)$ is meant to minimize any bias that the *sampling process* may introduce in the regression, and no change is necessary with the particular policy π^θ considered.

Given an estimate V_{v^*} of the value function associated with a given policy π^θ, the corresponding advantage function can now be approximated by solving the following regression problem:

$$w^* = \arg\min_w \sum_t \frac{1}{\hat{\mu}(x_t)}\big(r_t + \gamma V_{v^*}(y_t) - V_{v^*}(x_t) - \phi^\top(x_t,a_t)w\big)^2,$$

where each $\phi_i(x,a)$ is a compatible basis function verifying (2). We remark that no importance sampling is necessary in the above estimation, as can easily be seen by repeating the above computations for the advantage function. The regression problem can now easily be solved by setting

$$\mathbf{M} = \sum_t \frac{1}{\hat{\mu}(x_t)}\phi(x_t,a_t)\phi^\top(x_t,a_t)$$

and

$$\mathbf{b} = \sum_t \frac{\phi(x_t,a_t)}{\hat{\mu}(x_t)}\big(r_t + \gamma V_{v^*}(x_{t+1}) - V_{v^*}(x_{t+1})\big),$$

from where we obtain $w^* = \mathbf{M}^{-1}\mathbf{b}$. We conclude by observing that, with enough samples, the inverse in the expression above is well defined, since we assume the functions ϕ_i to be linearly independent.

3.4 Analysis and Discussion of the Algorithm

We now discuss several important properties of FNAC and compare it with other related algorithms in the literature.

We start by remarking that the general regression-based critic and the importance sampling "regularization" imply that *the dataset can be made independent of the current learning policy*. The main consequence of this is that, by requiring minimum regularity conditions from the regression algorithm, the following result can easily be established:

Theorem 3. *Let $\mathfrak{F}(\theta)$ denote the regression algorithm used in FNAC to produce the estimates $\hat{A}^\theta$ and $\hat{V}^\theta$ for an MDP $\mathcal{M}$, given a θ-dependent policy π^θ. Then, if $\mathfrak{F}$ is Lipschitz w.r.t. θ, and the step-size sequence $\{\alpha_t\}$ used in the actor update is such that*

$$\sum_t \alpha_t = \infty \qquad\qquad \sum_t \alpha_t^2 < \infty$$

FNAC converges with probability 1.

Proof. Due to space limitations, we provide only a brief sketch of the proof.

The result arises as a consequence of the convergence results in [18]. In the referred paper, convergence is established by analyzing a two-time-scale update of the general form:

$$\begin{aligned} w_{k+1} &= w_k + \alpha_k F(w_k, \theta_k) \\ \theta_{k+1} &= \theta_k + \beta_k G(w_{k+1}, \theta_k), \end{aligned} \tag{7}$$

where $\alpha = \mathrm{o}(\beta_k)$. By requiring several mild regularity conditions on the underlying process, on the policy space and on the set of basis functions, convergence can be established by an ODE argument. In particular, the proof starts by establishing global asymptotic stability of the "faster" ODE

$$\dot{w}_t = f(w_t, \theta),$$

for every θ, where f is an "averaged" version of F. Then, denoting by $w^*(\theta)$ the corresponding limit point, the proof proceeds by establishing the global asymptotic stability of the "slower" ODE

$$\dot{\theta}_t = g(w^*(\theta_t), \theta_t),$$

where, once again, g is an "averaged" version of G. Convergence of the coupled iteration (7) is established as long as $w^*(\theta)$ is Lipschitz continuous w.r.t. θ.

In terms of our analysis, and since our critic is a regression algorithm, it holds that at every iteration of the actor the critic *is actually in the corresponding limit* $w^*(\theta)$. Therefore, since our data is policy-independent and we assume our regression algorithm to be Lipschitz on θ, the convergence of our method is a consequence of the corresponding result in [18]. □

Another important difference that distinguished FNAC from other works is the efficient data use in regression methods. To emphasize this point, notice that after each policy update by the actor, the critic must evaluate the updated policy, computing the associated advantage function. In previous actor-critic algorithms [1, 15, 18], this required the acquisition of *new sampled data obtained using the updated policy.* However, in many problems, the acquisition of new data is a costly and time-consuming process. Furthermore, the evaluation of the updated policy makes poor use of previously used data.[6] In our algorithm, *all data collected* can be used in every iteration. Evidently, the importance-sampling term in (6) will weight some samples more than others in the estimation of each V^θ, but this is conducted so as to take full advantage of available data.

Finally, we remark that, from Theorem 2 and subsequent developments, the (natural) gradient is computed from the orthogonal projection of Q^θ/A^θ into $\mathcal{L}(\phi)$. However, as all other actor-critic methods currently available [1, 15, 17, 18], our critic cannot compute this projection exactly, since it would require the knowledge of the function to be projected. This will impact the performance of the algorithm in a similar way to that stated in Theorem 1 of [18].

4 Experimental Results

We conducted several experiments to evaluate the behavior of our algorithm in a simple continuous state, continuous action problem. We applied FNAC to the well-known mountain-car problem [27]. In this problem, an underpowered car must go up a steep hill, as depicted in Figure 2. As it has not enough acceleration to go all the way up the hill, it must bounce back-and-forth to gain enough velocity to climb up. The car is described by two continuous state-variables, namely position p and velocity v, and is controlled by a single continuous action, the acceleration a. The range of allowed positions is $[-1.2; +0.5]$ and the velocity ranges between -0.07 and 0.07. The acceleration takes values in the interval $[-1, 1]$. The car can be described by the dynamic equations:

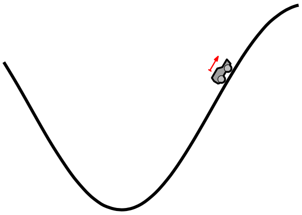

Fig. 2. The mountain-car problem: an underpowered car must go up a steep hill

$$v(t+1) = \mathsf{bound}\big[v(t) + 0.001a(t) - 0.0025\cos(3p(t))\big]$$
$$p(t+1) = \mathsf{bound}\big[p(t) + v(t+1)\big]$$

[6] In incremental algorithms [15, 18], the step-size sequence works as a “forgetting” factor.

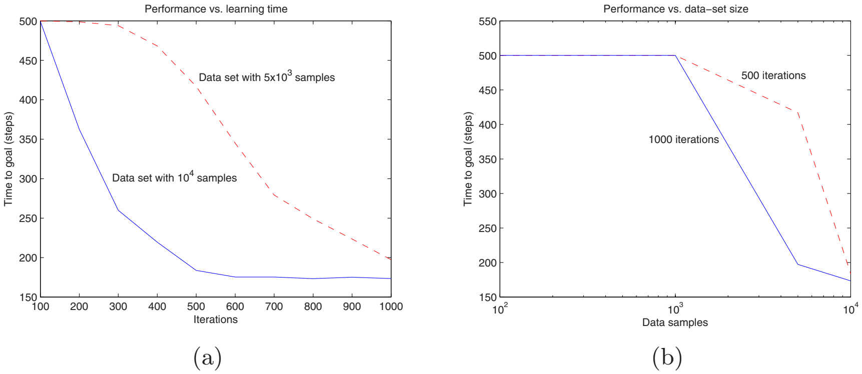

Fig. 3. Average performance of FNAC in the mountain-car problem: (a) Average time to reach the goal as the number of iterations is increased. (b) Average time to reach the goal as the size of the dataset is increased. The results depicted correspond to the average over 100 independent Monte-Carlo trials. Average times of 500 steps indicate that the car was not able to reach the goal.

where the function bound maintains the values of p and v within the limits. Whenever the car reaches the position limits, its velocity is set to zero. Whenever the car reaches the top of the hill, its position and velocity are randomly reset and the car gets a reward of 10. Otherwise, it gets a reward of -1.

We are interested in analyzing how much data and time are required for the FNAC algorithm to learn a "good" policy. To this purpose, we ran the FNAC algorithm with different dataset sizes and for different amounts of time. For each such run, we evaluated the learnt policy, initializing the car in the bottom-most position of the environment with zero velocity and running the learnt policy up to a maximum of 500 steps. We then observed the number of steps taken to reach the goal. Figure 3.a shows the evolution of the average time to goal as we increase the number of iterations of the algorithm.

It is clear from Figure 3.a that after only 300 iterations the algorithm already learnt a good policy (one that is able to reach the goal) and, for the case of a dataset of 10^4 points, the policy after 600 iterations is practically as good as the best policy computed. It is important to refer at this point that we used a simple linear approximator with 16 RBFs spread uniformly across the state space. The policy was parameterized using a Boltzmann-like distribution relying on the linear combination of 64 RBF uniformly distributed across the state-action space. Notice also that, by using a linear approximator, the regression in (6) can be computed analytically.

We also present in Figure 3.b the evolution of the average time to goal as we increase the size of the dataset. We tested the performance of the algorithm with datasets containing 100, 500, 10^3, 5×10^3 and 10^4 samples.[7] As clearly

[7] We notice that, due to the deterministic transitions, no double sampling is necessary in this particular example.

seen from the results in Figure 3.b, the size of the dataset greatly influences the performance of the algorithm. In the particular problem considered here, the FNAC algorithm was able to find a "good" policy with 5×10^3 points and the best performance was attained with a dataset of 10^4 samples.

5 Concluding Remarks

In this paper we presented the fitted natural actor-critic algorithm (FNAC). Our algorithm uses natural policy-gradient updates in the actor. However, unlike other natural actor-critic (NAC) algorithms, the critic component of FNAC relies on regression methods, with several important advantages:

– The use of regression methods allows the estimates of the *value-function* to use *general function approximation*, unlike previous NAC algorithms which, due to their use of TD-based critics, are limited to linear approximators. Notice, however, that Theorem 2 requires A^π to be linearly approximated, using a compatible set of basis functions verifying (2). The use of general function approximation for V^π will impact (in a positive way) the accuracy with which A^π can be represented.
– By adding an importance-sampling component, we allow our critic to *reuse all data* in all iterations of the algorithm, this being a fundamental advantage in problems where collecting data is costly or time consuming;
– The reuse of data allows the algorithm to consider datasets which are policy-independent. This, means that, unlike other NAC methods, the convergence of the algorithm can be conducted by a simple ODE argument;

It is also worth mentioning that FNAC is amenable to a multitude of different implementations, fully exploring the power of general fitting/regression methods.

The work portrayed here also suggests several interesting avenues for future research. First of all, and although not discussed in the paper, an online implementation of our algorithm can easily be obtained by an iterative implementation of the regression routines. Our convergence result holds if the relative weight of each sample in the data-set is stored (in terms of sampling policy).

Also, our initial experimental results illustrate the efficient use of data of our algorithm, since FNAC could attain good performance, reusing the same dataset at every iteration. Currently we are exploring the performance of the algorithm in more demanding tasks (namely, robotic grasping tasks encompassing high-dimensional state and action spaces). It would also be interesting to have some quantitative evaluation of the advantages of FNAC in face of other methods for MDPs with continuous state-action spaces. However, a direct comparison is not possible: in the current implementation of FNAC, the data gathering process is completely decoupled from the actual algorithm, while in most methods both processes occur simultaneously, thus impacting the corresponding learning times.

On a more general perspective, the critic component in FNAC estimates at each iteration the value function V^θ by minimizing the empirical Bellman residual. Approximations relying on Bellman residual minimization are more stable and more "predictable" than TD-based approaches [28]. It would be interesting to further explore these results to gain a deeper understanding of the advantages of this type of approximation in the setting considered here.

Finally, the simple and elegant results arising from the consideration of natural gradients suggests that it may be possible to further extend this approach and make use of higher-order derivatives (*e.g.*, as in Newton-like methods) to develop policy search methods for RL problems.

Acknowledgements

The authors would like to acknowledge the useful comments from Jan Peters and the anonymous reviewers.

References

1. Peters, J., Vijayakumar, S., Schaal, S.: Natural Actor-Critic. In: Proc. European Conf. Machine Learning, pp. 280–291 (2005)
2. Bertsekas, D., Tsitsiklis, J.: Neuro-Dynamic Programming. Athena Scientific (1996)
3. Tesauro, G.: TD-Gammon, a self-teaching backgammon program, achieves master-level play. Neural Computation 6(2), 215–219 (1994)
4. Baird, L.: Residual algorithms: Reinforcement learning with function approximation. In: Proc. Int. Conf. Machine Learning, pp. 30–37 (1995)
5. Tsitsiklis, J., Van Roy, B.: An analysis of temporal-difference learning with function approximation. IEEE Trans. Automatic Control 42(5), 674–690 (1996)
6. Sutton, R.: Open theoretical questions in reinforcement learning. In: Proc. European Conf. Computational Learning Theory, pp. 11–17 (1999)
7. Antos, A., Munos, R., Szepesvári, C.: Fitted Q-iteration in continuous action-space MDPs. In: Adv. Neural Information Proc. Systems, vol. 20 (2007)
8. Munos, R., Szepesvári, C.: Finite-time bounds for sampling-based fitted value iteration. J. Machine Learning Research (submitted, 2007)
9. Gordon, G.: Stable fitted reinforcement learning. In: Adv. Neural Information Proc. Systems, vol. 8, pp. 1052–1058 (1996)
10. Ormoneit, D., Sen, S.: Kernel-based reinforcement learning. Machine Learning 49, 161–178 (2002)
11. Ernst, D., Geurts, P., Wehenkel, L.: Tree-based batch mode reinforcement learning. J. Machine Learning Research 6, 503–556 (2005)
12. Riedmiller, M.: Neural fitted Q-iteration: First experiences with a data efficient neural reinforcement learning method. In: Gama, J., Camacho, R., Brazdil, P.B., Jorge, A.M., Torgo, L. (eds.) ECML 2005. LNCS (LNAI), vol. 3720, pp. 317–328. Springer, Heidelberg (2005)
13. Kimura, H., Kobayashi, S.: Reinforcement learning for continuous action using stochastic gradient ascent. In: Proc. Int. Conf. Intelligent Autonomous Systems, pp. 288–295 (1998)
14. Lazaric, A., Restelli, M., Bonarini, A.: Reinforcement learning in continuous action spaces through sequential Monte Carlo methods. In: Adv. Neural Information Proc. Systems, vol. 20 (2007)
15. Konda, V., Tsitsiklis, J.: On actor-critic algorithms. SIAM J. Control and Optimization 42(4), 1143–1166 (2003)

16. Barto, A., Sutton, R., Anderson, C.: Neuronlike adaptive elements that can solve difficult learning control problems. IEEE Trans. Systems, Man and Cybernetics 13(5), 834–846 (1983)
17. van Hasselt, H., Wiering, M.: Reinforcement learning in continuous action spaces. In: Proc. 2007 IEEE Symp. Approx. Dynamic Programming and Reinforcement Learning, pp. 272–279 (2007)
18. Bhatnagar, S., Sutton, R., Ghavamzadeh, M., Lee, M.: Incremental natural actor-critic algorithms. In: Adv. Neural Information Proc. Systems, vol. 20 (2007)
19. Kakade, S.: A natural policy gradient. In: Adv. Neural Information Proc. Systems, vol. 14, pp. 1531–1538 (2001)
20. Puterman, M.: Markov Decision Processes: Discrete Stochastic Dynamic Programming. John Wiley & Sons, Inc., Chichester (1994)
21. Sutton, R., McAllester, D., Singh, S., Mansour, Y.: Policy gradient methods for reinforcement learning with function approximation. In: Adv. Neural Information Proc. Systems, vol. 12, pp. 1057–1063 (2000)
22. Marbach, P., Tsitsiklis, J.: Simulation-based optimization of Markov reward processes. IEEE Trans. Automatic Control 46(2), 191–209 (2001)
23. Meyn, S., Tweedie, R.: Markov Chains and Stochastic Stability. Springer, Heidelberg (1993)
24. Baird, L.: Advantage updating. Tech. Rep. WL-TR-93-1146, Wright Laboratory, Wright-Patterson Air Force Base (1993)
25. Amari, S.: Natural gradient works efficiently in learning. Neural Computation 10(2), 251–276 (1998)
26. Antos, A., Szepesvári, C., Munos, R.: Learning near-optimal policies with Bellman-residual minimization based fitted policy iteration and a single sample path. Machine Learning 71, 89–129 (2008)
27. Singh, S., Sutton, R.: Reinforcement learning with replacing eligibility traces. Machine Learning 22, 123–158 (1996)
28. Munos, R.: Error bounds for approximate policy iteration. In: Proc. Int. Conf. Machine Learning, pp. 560–567 (2003)