

Similarities and differences between policy gradient methods and evolution strategies

Verena Heidrich-Meisner and Christian Igel *

Institut für Neuroinformatik, Ruhr-Universität Bochum, Germany

Abstract. Natural policy gradient methods and the covariance matrix adaptation evolution strategy, two variable metric methods proposed for solving reinforcement learning tasks, are contrasted to point out their conceptual similarities and differences. Experiments on the cart pole benchmark are conducted as a first attempt to compare their performance.

1 Introduction

Reinforcement learning (RL) algorithms search for a policy mapping states of the environment to (a probability distribution over) the actions an agent can take in those states. The goal is to find a behavior such that some notion of future reward is maximized. Direct policy search methods address this task by directly learning parameters of a function explicitly representing the policy. Here we consider two general approaches to conduct direct policy search, namely policy gradient methods (PGMs) and evolution strategies (ESs). We will argue that these approaches are quite similar. This makes it all the more surprising that so far there has been no systematic comparison of PGMs and ESs applied to the same test problems and operating on the same class of policies with the same parameterization. This paper is our attempt to draw such a comparison, on a conceptual level and by conducting first empirical studies. We restrict our consideration to the *natural actor critic* algorithm (NAC, [1, 2]) and the *covariance matrix adaptation ES* (CMA-ES, [3]), which are compared in the context of optimization in [4]. We picked these two because they can be considered state-of-the-art, they are our favorite direct policy search method and evolutionary RL algorithm, respectively, and they are both variable metric methods.

In section 2 we briefly review the NAC algorithm and the CMA-ES. Section 3 describes the conceptual relations of these two approaches and in section 4 we use a simple toy problem to compare the methods empirically.

2 Direct policy search methods

Although our methods are applicable in non-Markovian environments, for this section we assume that the RL problem is described by a Markov decision process $[\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}]$, where \mathcal{S} denotes the set of states, \mathcal{A} the possible actions, $\mathcal{P}_{s,s'}^a$ the probability that action a taken in state s leads to state s' , and $\mathcal{R}_{s,s'}^a$ the

*The authors acknowledge support from the German Federal Ministry of Education and Research within the Bernstein group “The grounding of higher brain function in dynamic neural fields”.

expected reward received after transition from state s to s' while performing action a . An agent interacts with the environment on a discrete time scale following a behavioral policy $\pi : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$, where $\pi(s, a)$ is the probability to choose action a in state s (for deterministic policies we write $\pi : \mathcal{S} \rightarrow \mathcal{A}$). The goal of reinforcement learning is to find a behavioral policy π such that some notion of expected future reward $\rho(\pi)$ is maximized. For example, for episodic tasks we can define $\rho(\pi) = \sum_{s, s' \in \mathcal{S}, a \in \mathcal{A}} d^\pi(s) \pi(s, a) \mathcal{P}_{s, s'}^a \mathcal{R}_{s, s'}^a$, where $d^\pi(s) = \lim_{t \rightarrow \infty} \Pr\{s_t = s \mid s_0, \pi\}$ is the stationary state distribution, which we assume to exist, s_t is state in time step t , and $r_{t+1} \in \mathbb{R}$ is the reward received after the action in time step t . Most RL algorithms learn value functions measuring the quality of an action in a state and define the policy on top of these functions. Direct policy search methods search for a good policy in a parametrized space of functions. They may build on estimated value functions (as policy gradient methods usually do), but this is not necessary (e.g., in evolution strategies). For a recent introduction to RL we refer to [5].

Policy gradient ascent Policy gradient methods assume a differentiable structure on a predefined class of stochastic policies and ascent the gradient of a performance measure. Let the performance $\rho(\pi)$ of the current policy with parameters θ be defined as above. The performance gradient $\nabla_{\theta} \rho(\pi)$ with respect to the policy parameters θ is estimated from interaction with the environment.

The policy gradient theorem [6] ensures that the performance gradient can be determined from unbiased estimates \hat{Q}^π and \hat{d}^π of state-action value function $Q^\pi(s, a) = \mathbb{E}[\sum_{t=0}^{\infty} r_{t+1} \mid \pi, s_0 = s, a_0 = a]$ and the stationary distribution, respectively. For any MDP we have $\nabla_{\theta} \rho = \sum_{s \in \mathcal{S}} d^\pi(s) \sum_{a \in \mathcal{A}} \nabla_{\theta} \pi(s, a) \hat{Q}^\pi(s, a)$. This formulation contains explicitly the unknown value function, which has to be estimated. It can be replaced by a function approximator $f_{\mathbf{w}} : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ (the *critic*) with real-valued parameter vector \mathbf{w} satisfying the *convergence condition* $\sum_{s \in \mathcal{S}} d^\pi(s) \sum_{a \in \mathcal{A}} \pi(s, a) [Q^\pi(s, a) - f_{\mathbf{w}}(s, a)] \nabla_{\mathbf{w}} f_{\mathbf{w}}(s, a) = 0$. This leads directly to the extension of the policy gradient theorem for function approximation. If $f_{\mathbf{w}}$ satisfies the convergence condition and is *compatible* with the policy parametrization in the sense that $\nabla_{\mathbf{w}} f_{\mathbf{w}}(s, a) = \nabla_{\theta} \pi(s, a) / \pi(s, a)$ (i.e., $f_{\mathbf{w}}$ is linear in the corresponding features), then the policy gradient theorem holds if $\hat{Q}^\pi(s, a)$ is replaced by $f_{\mathbf{w}}(s, a)$.

Stochastic policies π with parameters θ are parametrized probability distributions. In the space of probability distributions, the Fisher information matrix $F(\theta)$ induces an appropriate metric suggesting “natural” gradient ascent in the direction of $\tilde{\nabla}_{\theta} \rho(\pi) = F(\theta)^{-1} \nabla_{\theta} \rho(\pi)$. Using the definitions above, we have $F(\theta) = \sum_{s \in \mathcal{S}} d^\pi(s) \sum_{a \in \mathcal{A}} \pi_{\theta}(s, a) \nabla_{\theta} \ln(\pi(s, a)) (\nabla_{\theta} \ln(\pi(s, a)))^T$. This implies $\nabla_{\theta} \rho = F(\theta) \mathbf{w}$, which leads to the most interesting identity $\tilde{\nabla}_{\theta} \rho(\pi) = \mathbf{w}$. Bag-nell and Schneider [7] build an algorithm directly for the metric, while Peters et al. [1] integrate the natural gradient in an actor-critic architecture (NAC).

Evolution strategy Evolution strategies are direct iterative random search algorithms. In the CMA-ES [8, 3] a set of μ candidate solutions (here param-

eter vectors describing policies), the parent population, are maintained, from which $\lambda > \mu$ new candidate solutions are generated in each iteration. The performances of these offspring solutions (policies) are determined and the μ best form the parent population in the next iteration. In each iteration, the k th offspring $\mathbf{x}_k \in \mathbb{R}^n$ is generated by multi-variate *Gaussian mutation* and *weighted global intermediate recombination*, i.e., $\mathbf{x}_k = \langle \mathbf{x}_{\text{parents}} \rangle_{\mathbf{w}} + \sigma \mathbf{z}_k$, where $\mathbf{z}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{C})$ and $\langle \mathbf{x}_{\text{parents}} \rangle_{\mathbf{w}} = \sum_{i=1}^{\mu} w_i \mathbf{x}_{i\text{th-best-parent}}$ ($w_i \propto \ln(\mu + 1) - \ln(i)$, $\|\mathbf{w}\|_1 = 1$). The CMA-ES is a variable metric algorithm adapting both the n -dimensional covariance matrix \mathbf{C} of the normal mutation distribution as well as the *global step size* $\sigma \in \mathbb{R}^+$. In the basic algorithm, a low-pass filtered *evolution path* \mathbf{p} of successful directions (i.e., selected steps) is stored, $\mathbf{p} \leftarrow \eta_1 \mathbf{p} + \eta_2 (\langle \mathbf{x}_{\text{new parents}} \rangle - \langle \mathbf{x}_{\text{old parents}} \rangle)$, and \mathbf{C} is changed to make steps in the direction \mathbf{p} more likely: $\mathbf{C} \leftarrow \eta_3 \mathbf{C} + \eta_4 \mathbf{p} \mathbf{p}^T$ (this rank-one update of \mathbf{C} is augmented by a rank- μ update, see [8]). The variables η_1, \dots, η_4 denote fixed learning rates and normalization constants set to default values [8].

The global step size σ is adapted on a faster timescale. It is increased if the selected steps are larger and/or more correlated than expected and decreased if they are smaller and/or more anticorrelated than expected. The highly efficient use of information and the fast adaptation of σ and \mathbf{C} makes the CMA-ES one of the best direct search algorithms for real-valued optimization. It was successfully applied to RL using policies based on neural dynamics in [9].

3 Comparison of NAC and CMA-ES

Policy gradient methods and ESs are similar to each other in several respects, see Fig. 2. They search directly in policy space, thus the actor-part in the agent is learned actively. However, while ESs are actor-only methods, the NAC has an actor-critic architecture. In both approaches the class of possible policies is given by a parametrized family of functions, but in the case of PGMs the choice of the policy class is restricted by the compatibility condition of the policy gradient theorem.

Both PGMs and ESs rely on random perturbations to explore the space of

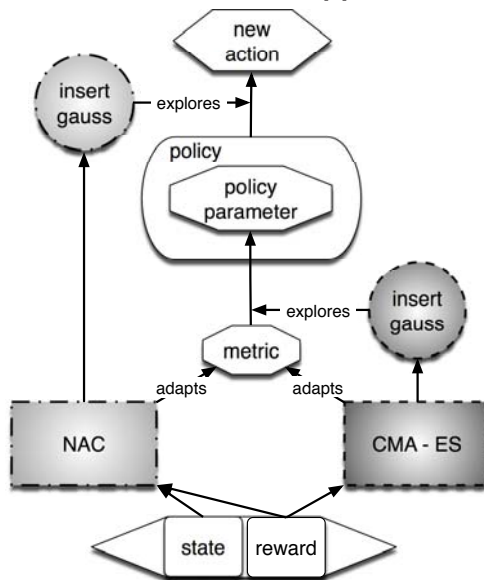


Fig. 1: Conceptual similarities and differences of natural policy gradient ascent and CMA evolution strategy: Both methods adapt a metric for the variation of the policy parameters based on information received from the environment. Both explore by stochastic perturbation of policies, but at different levels.

policies. Evolutionary methods usually perturb a deterministic policy by mutation and recombination, while in PGMs the random variations are an inherent property of the stochastic policies. While the number n of parameters of the policy determines the n -dimensional random variation in the CMA-ES, in the PGMs the usually lower dimensionality of the action corresponds to the dimensionality of the random perturbations. However, in ESs there is only one initial stochastic variation per episode, while the stochastic policy introduces perturbations in every step of the episode.

The CMA-ES as well as the NAC are variable-metric methods. A natural policy gradient method implicitly estimates the Fisher metric to follow the natural gradient of the performance in the space of the policy parameters and chooses its action according to a stochastic policy. Assuming a Gaussian distribution of the actions this resembles the CMA-ES. In the CMA-ES the parameters are perturbed according to a multi-variate Gaussian distribution. The covariance matrix of this distribution is adapted online. This corresponds to learning an appropriate metric for the optimization problem at hand. After the stochastic variation the actions are chosen deterministically. Thus, both types of algorithms perform the same conceptual steps to obtain the solution. They differ in the order of these steps and the level at which the random changes are applied.

4 Experiments

For our first experiments we focus on one single task, pole-balancing, which is a well-known benchmark problem in RL. A pole is mounted on a cart living in a 1-dimensional space; the goal is to balance the pole in the center of the available area as long as possible by applying forces to the cart. The problem is treated as an episodic task, where an episode either ends when the pole falls down (when the angle becomes larger than 0.7 rad), when the cart leaves the available area, or after a predefined maximum number of time steps N . The state description consists of current position and velocity of the cart and the current angle and angular velocity of the pole $\mathbf{s} = (x, \dot{x}, \zeta, \dot{\zeta})^T$. The agent receives a reward of 0 for every time step the agent is in the target area (space coordinate of the cart is close to the center $|x| < 0.05$ and the pole is almost perpendicular $|\zeta| < 0.05$), a reward of $-2(N - t)$, if the pole crashes ($|\zeta| > 0.7$) or leaves the allowed x -range ($|x| > 2.4$) at time step t , and a reward of -1 for every other time step. The starting point for an episode is drawn from one of two start intervals: $x \in [-2, 2]$, $\zeta \in [-0.6, 0.6]$ and $x \in [-0.2, 0.2]$, $\zeta \in [-0.2, 0.2]$. The velocities are always initialized with 0. Both methods operate on the same policy class $\pi_{\boldsymbol{\theta}}^{\text{deter}}(\mathbf{s}) = \boldsymbol{\theta}^T \mathbf{s}$ with $\mathbf{s}, \boldsymbol{\theta} \in \mathbb{R}^4$. For learning, the NAC uses the stochastic policy $\pi_{\boldsymbol{\theta}}^{\text{stoch}}(\mathbf{s}, a) = \text{N}(\pi_{\boldsymbol{\theta}}^{\text{deter}}(\mathbf{s}), \sigma_{\text{NAC}})$, where the variance σ_{NAC} is viewed as an additional adaptive parameter of the PGM. The NAC is evaluated on the corresponding deterministic policy. In all experiments the same number of $n_{\text{eval}} = 10$ episodes is used for assessing the performance of a policy. We analyse both learning from scratch where the initial policy is set to $\boldsymbol{\theta} = 0$ and from initial policies with an average performance of $\bar{\rho} = 376.1$ on the easiest set

