Variable Metric Reinforcement Learning Methods Applied to the Noisy Mountain Car Problem

Verena Heidrich-Meisner and Christian Igel

Institut für Neuroinformatik, Ruhr-Universität Bochum, Germany
{Verena.Heidrich-Meisner,Christian.Igel}@neuroinformatik.rub.de

Abstract. Two variable metric reinforcement learning methods, the natural actor-critic algorithm and the covariance matrix adaptation evolution strategy, are compared on a conceptual level and analysed experimentally on the mountain car benchmark task with and without noise.

1 Introduction

Reinforcement learning (RL) algorithms address problems where an agent is to learn a behavioural policy based on reward signals, which may be unspecific, sparse, delayed, and noisy. Many different approaches to RL exist, here we consider policy gradient methods (PGMs) and evolution strategies (ESs). This paper extends our previous work on analysing the conceptual similarities and differences between PGMs and ESs [1].

For the time being, we look at single representatives of each approach that have been very successful in their respective area, the *natural actor critic* algorithm (NAC, [2–5]) and the *covariance matrix adaptation ES* (CMA-ES, [6]). Both are variable metric methods, actively learning about the structure of the search space. The CMA-ES is regarded as state-of-the-art in real-valued evolutionary optimisation [7]. It has been successfully applied and compared to other methods in the domain of RL [8–12]. Interestingly, recent studies compare CMA-ES and variants of the NAC algorithm in the context of optimisation [13], while we look at both methods in RL.

We promote the CMA-ES for RL because of its efficiency and, even more important, its robustness. The superior robustness compared to other RL algorithms has several reasons, but probably the most important reason is that the adaptation of the policy as well as of the metric is based on ranking policies, which is much less error prone than estimating absolute performance or performance gradients.

Our previous comparison of NAC and CMA-ES on different variants of the single pole balancing benchmark in [1] indicate that the CMA-ES is more robust w.r.t. to the choice of hyperparameters (such as initial learning rates) and initial policies compared to the NAC. In [1] the NAC performed on par with the CMA-ES in terms of learning speed only when fine-tuning policies, but worse for

harder pole balancing scenarios. In this paper, we compare the two methods applied to the mountain car problem [14] to support our hypotheses and previous findings. As the considered class of policies has only two parameters, this benchmark serves as some kind of minimal working example for RL methods learning correlations between parameters. The performance of random search provides a performance baseline in our study. In order to investigate the robustness of the algorithms, we study the influence of noise added to the observations.

The paper is organised as follows. In section 2 we review the NAC algorithm and the CMA-ES for RL. Section 3 describes the conceptual relations of these two approaches and in section 4 we empirically compare the methods.

2 Reinforcement learning directly in policy space

Markov decision processes (MDP) are the basic formalism to describe RL problems. An MDP $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R} \rangle$ consists of the set of states $\mathcal{S}$, the possible actions $\mathcal{A}$, the probabilities $\mathcal{P}^a_{s,s'}$ that an action a taken in state s leads to state s', and the expected rewards $\mathcal{R}^a_{s,s'}$ received when going from state a s to s' after performing an action a.

Partially observable Markov decision processes (POMDP) are a generalisation of MDPs [15]. In a POMDP, the environment is determined by an MDP, but the agent cannot directly observe the state of the MPD. Formally, a POMDP can be described as a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \Omega, O \rangle$. The first four elements define the underlying MDP, Ω is the set of observations an agent can perceive, and the observation function $S : \mathcal{S} \times \mathcal{A} \to \Lambda(\Omega)$ maps a state and the action that resulted in this state to a probability distribution over observations (i.e., $S(s', a)(o)$ is the probability of observing o given that the agent took action a and landed in state s').

The goal of RL 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 $\rho(\pi) = \sum_{s,s' \in \mathcal{S}, a \in \mathcal{A}} d^\pi(s)\pi(s,a)\mathcal{P}^a_{s,s'}\mathcal{R}^a_{s,s'}$, where $d^\pi(s) = \sum_{t=0}^{\infty} \gamma^t \Pr\{s_t = s \mid s_0, \pi\}$ is the stationary state distribution, which we assume to exist, s_t is state in time step t, and $\gamma \in]0, 1]$ a discount parameter. The immediate reward received after the action in time step t is denoted by $r_{t+1} \in \mathbb{R}$.

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 and PGMs search for a good policy in a parametrised space of functions. They may build on estimated value functions (as PGMs usually do), but this is not necessary (e.g., in ESs).

2.1 Natural policy gradient ascent

Policy gradient methods operate on a predefined class of stochastic policies. They require a differentiable structure to ensure the existence of the gradient of the performance measure and ascent this gradient. Let the performance $\rho(\pi)$ of the current policy with parameters $\boldsymbol{\theta}$ be defined as above. Because in general neither

d^π, $\mathcal{R}$, nor $\mathcal{P}$ are known, the performance gradient $\boldsymbol{\nabla}_{\boldsymbol{\theta}}\rho(\pi)$ with respect to the policy parameters $\boldsymbol{\theta}$ is estimated from interaction with the environment.

The policy gradient theorem [16] ensures that the performance gradient can be determined from unbiased estimates of the state-action value function $Q^\pi(s,a) = \mathbb{E}\left[\sum_{t=0}^{\infty}\gamma^t r_{t+1}|\pi, s_0 = s, a_0 = a\right]$ and stationary distribution, respectively. For any MDP we have

$$\boldsymbol{\nabla}_{\boldsymbol{\theta}}\rho = \sum_{s\in\mathcal{S}} d^\pi(s) \sum_{a\in\mathcal{A}} \boldsymbol{\nabla}_{\boldsymbol{\theta}}\pi(s,a) Q^\pi(s,a) \ . \tag{1}$$

This formulation contains explicitly the unknown value function, which has to be estimated. It can be replaced by a function approximator $f_{\boldsymbol{v}} : \mathcal{S}\times\mathcal{A}\to\mathbb{R}$ (the *critic*) with real-valued parameter vector $\boldsymbol{v}$ satisfying the *convergence condition* $\sum_{s\in\mathcal{S}} d^\pi(s)\sum_{a\in\mathcal{A}}\pi(s,a)\left[Q^\pi(s,a) - f_{\boldsymbol{v}}(s,a)\right]\boldsymbol{\nabla}_{\boldsymbol{v}} f_{\boldsymbol{v}}(s,a) = 0$. This leads directly to the extension of the policy gradient theorem for function approximation. If $f_{\boldsymbol{v}}$ satisfies the convergence condition and is *compatible* with the policy parametrisation in the sense that $\boldsymbol{\nabla}_{\boldsymbol{v}} f_{\boldsymbol{v}}(s,a) = \boldsymbol{\nabla}_{\boldsymbol{\theta}}\pi(s,a)/\pi(s,a)$, that is,

$$f_{\boldsymbol{v}} = \boldsymbol{\nabla}_{\boldsymbol{\theta}} \ln(\pi(s,a))\boldsymbol{v} + \text{const} \ , \tag{2}$$

then the policy gradient theorem holds if $Q^\pi(s,a)$ in equation 1 is replaced by $f_{\boldsymbol{v}}(s,a)$ [16].

Stochastic policies π with parameters $\boldsymbol{\theta}$ are parametrised probability distributions. In the space of probability distributions, the Fisher information matrix $F(\boldsymbol{\theta})$ induces an appropriate metric suggesting "natural" gradient ascent in the direction of $\tilde{\boldsymbol{\nabla}}_{\boldsymbol{\theta}}\rho(\pi) = F(\boldsymbol{\theta})^{-1}\boldsymbol{\nabla}_{\boldsymbol{\theta}}\rho(\pi)$. Using the definitions above, we have

$$F(\boldsymbol{\theta}) = \sum_{s\in\mathcal{S}} d^\pi(s) \sum_{a\in\mathcal{A}} \pi(s,a)\boldsymbol{\nabla}_{\boldsymbol{\theta}} \ln(\pi(s,a))(\boldsymbol{\nabla}_{\boldsymbol{\theta}} \ln(\pi(s,a)))^{\mathrm{T}} \ .$$

This implies $\boldsymbol{\nabla}_{\boldsymbol{\theta}}\rho = F(\boldsymbol{\theta})\boldsymbol{v}$, which leads to the most interesting identity

$$\tilde{\boldsymbol{\nabla}}_{\boldsymbol{\theta}}\rho(\pi) = \boldsymbol{v} \ .$$

In the following, we derive the NAC according to [4, 5]. The function approximator $f_{\boldsymbol{v}}$ estimates the advantage function $A^\pi(s,a) = Q^\pi(s,a) - V^\pi(s)$, where $V^\pi(s) = \mathbb{E}\left[\sum_{t=0}^{\infty}\gamma^t r_{t+1}|\pi, s_0 = s\right]$ is the state value function. Inserting this in the Bellman equation for Q^π leads to

$$Q^\pi(s_t,a_t) = A^\pi(s_t,a_t) + V^\pi(s_t) = \sum_{s'} P^{a_t}_{s_t,s'}\left(\mathcal{R}^{a_t}_{s_t,s'} + \gamma V^\pi(s')\right) \ . \tag{3}$$

Now we insert equation 2 for the advantage function and sum up equation 3 over a sample path:

$$\sum_{t=0}^{T}\gamma^t A^\pi(s_t,a_t) = \sum_{t=0}^{T}\gamma^t r_{t+1} + \gamma^{T+1}V^\pi(s_{T+1}) - V(s_0) \ .$$

Algorithm 1: episodic Natural Actor-Critic

initialise $\boldsymbol{\theta} = \mathbf{0} \in \mathbb{R}^n$, $\boldsymbol{\Phi} = \mathbf{0} \in \mathbb{R}^{e_{\max} \times n+1}$, $\boldsymbol{R} = \mathbf{0} \in \mathbb{R}^{e_{\max}}$
for $k = 1, \ldots$ **do**
  // k counts number of policy updates
  for $e = 1, \ldots e_{\max}$ **do**
    // e counts number of episodes per policy update, $e_{\max} > n$
    for $t = 1, \ldots t_{\max}$ **do**
      // t counts number of time steps per episode
      begin
        observe state $\boldsymbol{s}_t$
        choose action a_t from $\boldsymbol{\pi}_{\boldsymbol{\theta}}$
        perform action a_t
        observe reward r_{t+1}
      end
      for $i = 1, \ldots, n$ **do**
        $[\boldsymbol{\Phi}]_{e,i} \leftarrow [\boldsymbol{\Phi}]_{e,i} + \gamma^t \frac{\partial}{\partial \theta_i} \ln \boldsymbol{\pi}_{\boldsymbol{\theta}}(s_t, a_t)$
      $[\boldsymbol{R}]_e \leftarrow [\boldsymbol{R}]_e + \gamma^t r_{t+1}$
    $[\boldsymbol{\Phi}]_{e,n+1} \leftarrow 1$
  // update policy parameters:
  $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + (\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{R}$

For an episodic task terminating in time step T it holds $V^\pi(s_{T+1}) = 0$. Thus, we have after replacing A^π with its approximation according to equation 2:

$$\sum_{t=0}^{T} \gamma^t (\boldsymbol{\nabla}_{\boldsymbol{\theta}} \ln \pi(s_t, a_t))^{\mathrm{T}} \boldsymbol{v} - V(s_0) = \sum_{t=0}^{T} \gamma^t r_{t+1}$$

For fixed start states we have $V^\pi(s_0) = \rho(\pi)$ and this is a linear regression problem with $n+1$ unknown variables $\boldsymbol{w} = [\boldsymbol{v}^{\mathrm{T}}, V^\pi(s_0)]^{\mathrm{T}}$ that can be solved after $n+1$ observed episodes (where n is the dimension of $\boldsymbol{\theta}$ and $\boldsymbol{v}$):

$$\begin{bmatrix} \sum_{t=0}^{T(e_1)} \left[\gamma^t \boldsymbol{\nabla}_{\boldsymbol{\theta}} \ln \pi(s_t^{e_1}, a_t^{e_1})\right]^{\mathrm{T}}, -1 \end{bmatrix}^{\mathrm{T}} \boldsymbol{v} = \sum_{t=0}^{T(e_1)} \gamma^t r_{t+1}^{e_1}$$
$$\vdots \qquad\qquad\qquad \vdots$$
$$\begin{bmatrix} \sum_{t=0}^{T(e_n)} \left[\gamma^t \boldsymbol{\nabla}_{\boldsymbol{\theta}} \ln \pi(s_t^{e_n}, a_t^{e_n})\right]^{\mathrm{T}}, -1 \end{bmatrix}^{\mathrm{T}} \boldsymbol{v} = \sum_{t=0}^{T(e_n)} \gamma^t r_{t+1}^{e_n}$$

The superscripts indicate the episodes. In algorithm 1 the likelihood information for a sufficient number of episodes is collected in a matrix $\boldsymbol{\Phi}$ and the return for each episode in $\boldsymbol{R}$. In every update step one inversion of the matrix $\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}$ is necessary.

2.2 Covariance matrix adaptation evolution strategy

We consider ESs for real-valued optimisation [17–20]. Let the optimisation problem be defined by an objective function $f : \mathbb{R}^n \to \mathbb{R}$ to be minimised, where n denotes the dimensionality of the search space (space of candidate solutions, decision space). Evolution strategies are random search methods, which iteratively sample a set of candidate solutions from a probability distribution over the search space, evaluate these points using f, and construct a new probability distribution over the search space based on the gathered information. In ESs, this search distribution is parametrised by a set of candidate solutions, the *parent population* with size μ, and by parameters of the variation operators that are used to create new candidate solutions (the *offspring population* with size λ) from the parent population.

In each iteration k, the lth offspring $\boldsymbol{x}_l \in \mathbb{R}^n$ is generated by multi-variate *Gaussian mutation* and *weighted global intermediate recombination*, i.e.,

$$\boldsymbol{x}_l^{(k+1)} = \left\langle \boldsymbol{x}_{\text{parents}}^{(k)} \right\rangle_{\boldsymbol{w}} + \sigma^{(k)} \boldsymbol{z}_l^{(k)} \ ,$$

where $\boldsymbol{z}_l^{(k)} \sim \mathrm{N}(\boldsymbol{0}, \boldsymbol{C}^{(k)})$ and $\left\langle \boldsymbol{x}_{\text{parents}}^{(k)} \right\rangle_{\boldsymbol{w}} = \sum_{i=1}^{\mu} w_i \boldsymbol{x}_{i\text{th-best-parent}}^{(k)}$ (a common choice is $\boldsymbol{w}_i \propto \ln(\mu+1) - \ln(i)$, $\|\boldsymbol{w}\|_1 = 1$).

The CMA-ES, shown in algorithm 2, is a variable metric algorithm adapting both the n-dimensional covariance matrix $\boldsymbol{C}^{(k)}$ of the normal mutation distribution as well as the *global step size* $\sigma^{(k)} \in \mathbb{R}^+$. In the basic algorithm, a low-pass filtered *evolution path* $\boldsymbol{p}^{(k)}$ of successful (i.e., selected) steps is stored,

$$\boldsymbol{p}_c^{(k+1)} \leftarrow (1 - c_c)\, \boldsymbol{p}_c^{(k)} + \sqrt{(c_c(2 - c_c)\mu_{\text{eff}})}\, \frac{1}{\sigma^{(k)}} \left(\left\langle \boldsymbol{x}_{\text{parents}}^{(k+1)} \right\rangle - \left\langle \boldsymbol{x}_{\text{parents}}^{(k)} \right\rangle \right),$$

and $\boldsymbol{C}^{(k)}$ is changed to make steps in the promising direction $\boldsymbol{p}^{(k+1)}$ more likely:

$$\boldsymbol{C}^{(k+1)} \leftarrow (1 - c_{\text{cov}})\, \boldsymbol{C}^{(k)} + c_{\text{cov}}\, \boldsymbol{p}_c^{(k+1)} \boldsymbol{p}_c^{(k+1)\,\mathrm{T}}$$

(this rank-one update of $\boldsymbol{C}^{(k)}$ can be augmented by a rank-μ update, see [21]). The variables c_c and c_{cov} denote fixed learning rates. The learning rate $c_{\text{cov}} = \frac{2}{(n+\sqrt{2})^2}$ is roughly inversely proportional to the degrees of freedom of the covariance matrix. The backward time horizon of the cumulation process is approximately c_c^{-1}, with $c_c = 4/(n+4)$ linear in the dimension of the path vector. Too small values for c_c would require an undesirable reduction of the learning rate for the covariance matrix. The *variance effective selection mass* $\mu_{\text{eff}} = \left(\sum_{t=1}^{\mu} w_i^2\right)^{-1}$ is a normalisation constant.

The global step size $\sigma^{(k)}$ 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:

$$\sigma^{(k+1)} \leftarrow \sigma^{(k)} \exp\left(\frac{c_\sigma}{d_\sigma} \left(\frac{\|\boldsymbol{p}_\sigma^{(k+1)}\|}{E\|\mathrm{N}(\boldsymbol{0}, \boldsymbol{I})\|} - 1 \right) \right) \ ,$$

Algorithm 2: rank-one CMA-ES

initialise $\boldsymbol{m}^{(0)} = \boldsymbol{\theta}$ and $\sigma^{(0)}$, evolution path $\boldsymbol{p}_\sigma^{(0)} = 0, \boldsymbol{p}_c^{(0)} = 0$ and covariance matrix $\boldsymbol{C}^{(0)} = \boldsymbol{I}$ (unity matrix)
for $k = 1, \ldots$ **do**
  // k counts number generations respective of policy updates
  for $l = 1, \ldots, \lambda$ **do**
    $x_l^{(k+1)} \sim \mathrm{N}(\boldsymbol{m}^{(k)}, {\sigma^{(k)}}^2 \boldsymbol{C}^{(k)})$ // create new offspring
  // evaluate offspring:
  for $l = 1, \ldots, \lambda$ **do**
    $f_l \leftarrow 0$ // fitness of lth offspring
    for $e = 1, \ldots e_{max}$ **do**
      // counts number of episodes per policy update
      for $t = 1, \ldots t_{max}$ **do**
        // t counts number of time steps per episode
        begin
          observe state $\boldsymbol{s}_t$,
          choose action a_t from π_θ,
          perform action a_t,
          observe reward r_{t+1}
        end
        $f_l \leftarrow f_l + \gamma^{t-1} r_{t+1}$
  // selection and recombination:
  $\boldsymbol{m}^{(k+1)} \leftarrow \sum_{i=1}^{\mu} w_i \boldsymbol{x}_{i:\lambda}^{(k)}$
  // step size control:
  $\boldsymbol{p}_\sigma^{(k+1)} \leftarrow (1 - c_\sigma)\boldsymbol{p}_\sigma^{(k)} + \sqrt{c_\sigma(2 - c_\sigma)\mu_{\mathrm{eff}}}\, {\boldsymbol{C}^{(k)}}^{-\frac{1}{2}} \frac{\boldsymbol{m}^{(k+1)} - \boldsymbol{m}^{(k)}}{\sigma^{(k)}}$
  $\sigma^{(k+1)} \leftarrow \sigma^{(k)} \exp \frac{c_\sigma}{d_\sigma} \left(\frac{\|\boldsymbol{p}_\sigma^{(k+1)}\|}{E\|\mathrm{N}(\boldsymbol{0}, \boldsymbol{I})\|} - 1 \right)$
  // covariance matrix update:
  $\boldsymbol{p}_c^{(k+1)} \leftarrow (1 - c_c)\boldsymbol{p}_c^{(k)} + \sqrt{c_c(2 - c_c)\mu_{\mathrm{eff}}} \frac{\boldsymbol{m}^{(k+1)} - \boldsymbol{m}^{(k)}}{\sigma^{(k)}}$
  $\boldsymbol{C}^{(k+1)} \leftarrow (1 - c_{\mathrm{cov}})\boldsymbol{C}^{(k)} + c_{\mathrm{cov}} \boldsymbol{p}_c^{(k+1)} {\boldsymbol{p}_c^{(k+1)}}^{\mathrm{T}}$

and its (*conjugate*) evolutions path is:

$$\boldsymbol{p}_s^{(k+1)} \leftarrow (1 - c_\sigma)\, \boldsymbol{p}_s^{(k)} + \sqrt{c_\sigma(2 - c_\sigma)\mu_{\mathrm{eff}}}\, {\boldsymbol{C}^{(k)}}^{-\frac{1}{2}} \left(\left\langle \boldsymbol{x}_{\mathrm{parents}}^{(k+1)} \right\rangle - \left\langle \boldsymbol{x}_{\mathrm{parents}}^{(k)} \right\rangle \right)$$

Again, $c_\sigma = \frac{\mu_{\mathrm{eff}}+2}{n+\mu_{\mathrm{eff}}+3}$ is a fixed learning rate and $d_\sigma = 1+2\max\left(0, \sqrt{\frac{\mu_{\mathrm{eff}}-1}{n+1}}\right)+c_\sigma$ a damping factor. The matrix $\boldsymbol{C}^{-\frac{1}{2}}$ is defined as $\boldsymbol{B}\boldsymbol{D}^{-1}\boldsymbol{B}^{\mathrm{T}}$, where $\boldsymbol{B}\boldsymbol{D}^2\boldsymbol{B}^{\mathrm{T}}$ is an eigendecomposition of $\boldsymbol{C}$ ($\boldsymbol{B}$ is an orthogonal matrix with the eigenvectors of $\boldsymbol{C}$ and $\boldsymbol{D}$ a diagonal matrix with the corresponding eigenvalues) and sampling $\mathrm{N}(\boldsymbol{0}, \boldsymbol{C})$ is done by sampling $\boldsymbol{B}\boldsymbol{D}\mathrm{N}(\boldsymbol{0}, \boldsymbol{I})$.

The values of the learning rates and the damping factor are well considered and have been validated by experiments on many basic test functions [21]. *They*

need not be adjusted dependent on the problem and are therefore no *hyperparameters of the algorithm.* Also the population sizes can be set to default values, which are $\lambda = \max(4 + \lfloor 3 \ln n \rfloor, 5)$ and $\mu = \lfloor \frac{\lambda}{2} \rfloor$ for offspring and parent population, respectively [21]. If we fix $\boldsymbol{C}^{(0)} = \boldsymbol{I}$, the only (also adaptive) hyperparameter that has to be chosen problem dependent is the initial global step size $\sigma^{(0)}$.

The CMA-ES uses rank-based selection. The best μ of the λ offspring form the next parent population.

The highly efficient use of information and the fast adaptation of σ and $\boldsymbol{C}$ makes the CMA-ES one of the best direct search algorithms for real-valued optimisation [7]. For a detailed description of the CMA-ES see the articles by Hansen et al. [22, 21, 6, 23].

3 Similarities and differences of NAC and CMA-ES

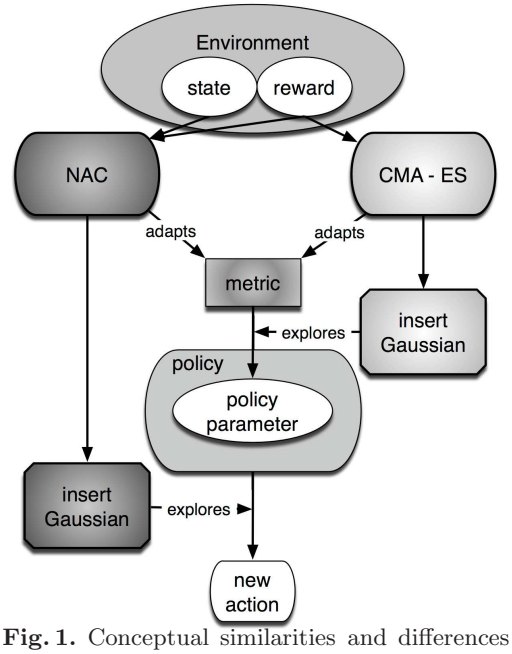

Fig. 1. Conceptual similarities and differences of natural policy gradient ascent and CMA evolution strategy: Both methods adapt a metric for the variation of the policy parameters based on information received from the environment. Both explore by stochastic perturbation of policies, but at different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

Policy gradient methods and ESs share several constituting aspects, see Fig. 1. Both search directly in policy space, thus the actor-part in the agent is represented and learnt actively. Yet, 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 parametrised family of functions, but in the case of PGMs the choice of the policy class is restricted to differentiable functions.

Exploration of the search space is realised by random perturbations in both ESs and PGMs. 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. In ESs there is only one initial stochastic variation per policy update. In contrast, the stochastic policy introduces perturbations in every step of the episode. 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. In ESs the search is driven solely by ranking policies and not by

the exact values of performance estimates or their gradients. The reduced number of random events and the rank-based evaluation are decisive differences and we hypothesise that they allow ESs to be more robust.

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 optimisation 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.

Policy gradient methods have the common properties of gradient techniques. They are powerful local search methods and thus benefit from a starting point close to an optimum. However, they are susceptible to being trapped in undesired local minima.

4 Experiments

The experiments conducted in this paper extend our previous work described in [1]. We have chosen the mountain car problem, which is a well-known benchmark problem in RL requiring few policy parameters.

The objective of this task is to navigate an underpowered car from a valley to a hilltop. The state $\boldsymbol{s}$ of the system is given by the position $x \in [-1.2, 0.6]$ of the car and by its current velocity $v = \dot{x} \in [-0.07, 0.07]$, actions are discrete forces applied to the car $a \in \{-a_{\max}, 0, a_{\max}\}$, where $a_{\max}$ is chosen to be insufficient to drive the car directly uphill from the starting position in the valley to the goal at the top. The agent receives a negative reward of $r = -1$ for every time step. An episode terminates when the car reaches the position $x = 0.5$, the discount parameter is set to $\gamma = 1$.

To allow for a fair comparison, both methods operate on the same policy class $\pi^{\text{deter}}_{\boldsymbol{\theta}}(\boldsymbol{s}) = \boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{s}$ with $\boldsymbol{s}, \boldsymbol{\theta} \in \mathbb{R}^2$ The continuous output a_{cont} of the policy is mapped by the environment to a discrete action $a \in \mathcal{A}$: $a = 1$ if $a_{\text{cont}} > 0.1$, $a = -1$ if $a_{\text{cont}} < 0.1$, and $a = 0$ otherwise. We also considered the mountain car problem with continuous actions, which, however, makes the task easier for the CMA-ES and more difficult for the NAC. For learning, the NAC uses the stochastic policy $\pi^{\text{stoch}}_{\boldsymbol{\theta}}(\boldsymbol{s}, a) = \mathrm{N}(\pi^{\text{deter}}_{\theta}(\boldsymbol{s}), \sigma_{\text{NAC}})$, 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 $e_{\max} = 10$ episodes is used for assessing the performance of a policy. We analyse two sets of start policies: $\boldsymbol{\theta} = \mathbf{0}$ (referred to as P_0) and drawing the components of $\boldsymbol{\theta}$ uniformly from $[-100., 100]$ (termed P_{100}). P_0 lies reasonably close to the

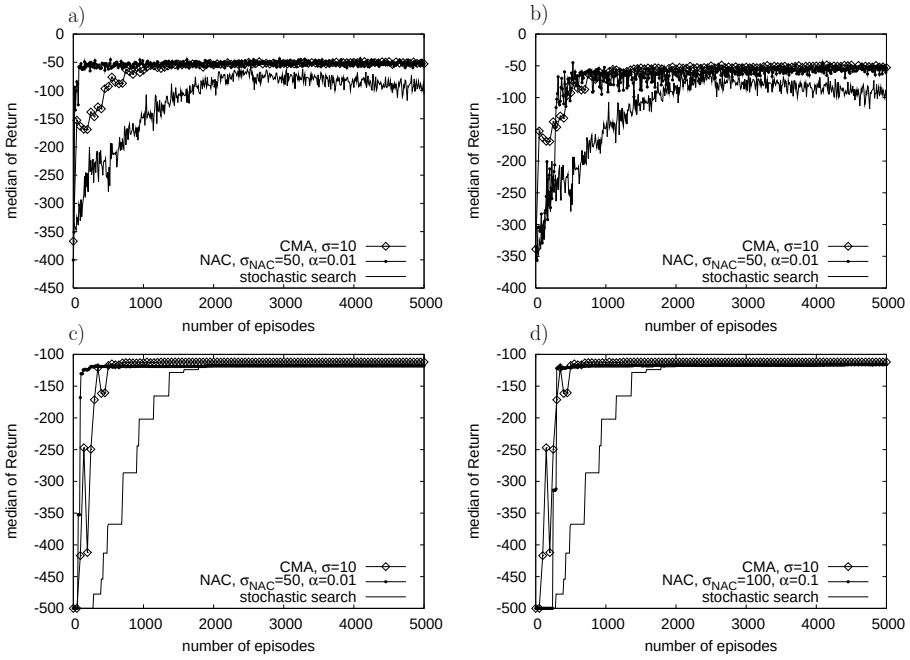

Fig. 2. Performance of NAC and CMA-ES on the mountain car task without noise based on 20 trials. a) CMA-ES, NAC, and stochastic search for initial policy P_0 and initial environment state S_{random} (with best respective parameter values) without noise b) CMA-ES, NAC, and stochastic search for initial policy P_{100} and initial environment state S_{random} (with best respective parameter values) without noise c) CMA-ES, NAC, and stochastic search for initial policy P_0 and initial environment state S_{fixed} (with best respective parameter values) without noise d) CMA-ES, NAC, and stochastic search for initial policy P_{100} and initial environment state S_{fixed} (with best respective parameter values) without noise

optimal parameter values. In the original mountain car task the start states for each episode are drawn randomly from the complete state space $\mathcal{S}$ (S_{random}). We additionally analyse the case (S_{fixed}) where all episodes start in the same state with position $x = -0.8$ and velocity $v = 0.01$. Driving simply in the direction of the goal is not sufficient to solve the problem for this starting condition.

As a baseline comparison we considered stochastic search, where policy parameters were drawn uniformly at random from a fixed interval and were then evaluated in the same way as CMA-ES and NAC.

In a second set of experiments we add Gaussian noise with zero mean and variance $\sigma_{\text{noise}} = 0.01$ to state observations (i.e., now we consider a POMDP).

Mountain car task without noise. Figure 2 shows the performance of NAC and CMA-ES on the mountain car problem. In the easiest cases (P_0 with S_{random} and

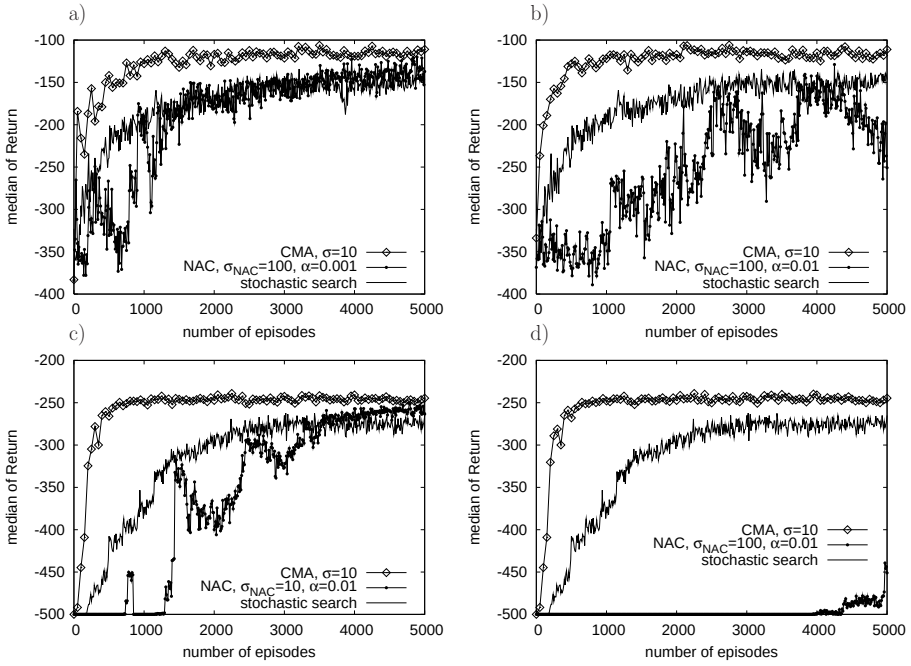

Fig. 3. Performance of NAC and CMA-ES on the mountain car task with noisy observations based on 20 trials. a) CMA-ES, NAC, and stochastic search for initial policy P_0 and initial environment state S_{random} (with best respective parameter values) with noise b) CMA-ES, NAC, and stochastic search for initial policy P_{100} and initial environment state S_{random} (with best respective parameter values) with noise c) CMA-ES, NAC, and stochastic search for initial policy P_0 and initial environment state S_{fixed} (with best respective parameter values) with noise d) CMA-ES, NAC, and stochastic search for initial policy P_{100} and initial environment state S_{fixed} (with best respective parameter values) with noise

S_{fixed}) the NAC clearly outperforms the CMA-ES. Here the NAC is also robust w.r.t changes of its hyperparameters (learning rate and initial variance). But this changes when the policy is not initialised close to the optimal parameter values and the parameters are instead drawn randomly. The CMA-ES performs as in the former case, but now it is faster than the NAC in the beginning. The NAC still reaches an optimal solution faster, but it is no longer robust. The CMA-ES is more stable in all cases. Its performance does not depend on the choice of initial policy at all and changing the value of its single parameter σ as the initial step size only marginally effects the performance, see figure 4 and tables 1 and 2.

Mountain car task with noise. For the next set of experiments we added noise to the observed state, thus creating a more realistic situation, see figure 3. In

Table 1. Final performance values of NAC on the mountain car task without noise with initial policy P_0 and starting states from S_{random} after 5000 episodes. The medians of 20 trials are reported.

α	0.0001	0.001	0.0001	0.0001	0.001	0.01	0.01	0.001
σ_{NAC}	10	10	100	50	100	1	10	1
final value	−49.4	−49.4	−49.55	−49.55	−50.7	−50.75	−50.85	−51.4
α	0.001	0.01	0.01	0.1	0.1	0.1	0.1	0.0001
σ_{NAC}	50	50	100	10	100	1	50	1
final value	−51.45	−52.35	−53.6	−73.3	−82.2	−98.55	−131.6	−147.45

Table 2. Final performance values of NAC on the mountain car task without noise with initial policy P_{100} and starting states from S_{random} after 5000 epsisodes.

α	0.001	0.01	0.01	0.001	0.1	0.01	0.1	0.1
σ_{NAC}	100	50	100	50	50	10	100	10
final value	−53.2	−54.25	−54.4	−59.15	−70.7	−84.45	−113.1	−117.25
α	0.0001	0.01	0.1	0.0001	0.0001	0.001	0.001	0.0001
σ_{NAC}	100	1	1	50	10	10	1	1
final value	−200.4	−289.15	−307.35	−309.2	−314.85	−316.1	−352.05	−354

Table 3. Final performance values of NAC on the mountain car task with noisy observations with initial policy P_0 and starting states from S_{random} after 5000 epsisodes.

α	0.001	0.001	0.01	0.0001	0.001	0.001	0.01	0.01
σ_{NAC}	50	100	100	100	10	1	1	50
final value	−133.17	−137.16	−137.83	−144.56	−164.71	−171.69	−174.34	−178.76
α	0.01	0.0001	0.0001	0.1	0.1	0.0001	0.1	0.1
σ_{NAC}	10	10	50	10	50	1	100	1
final value	−184.03	−185.15	−201.84	−336.33	−353.91	−377.36	−377.66	−380.71

Table 4. Final performance values of NAC on the mountain car task with noisy observations with initial policy P_{100} and starting states from S_{random} after 5000 epsisodes.

α	0.01	0.001	0.001	0.01	0.1	0.0001	0.01	0.0001
σ_{NAC}	100	100	50	50	50	1	1	100
final value	−131.81	−182.24	−212.23	−250.83	−308.37	−343.58	−349.77	−350.2
α	0.01	0.0001	0.001	0.1	0.1	0.0001	0.01	0.1
σ_{NAC}	10	50	1	10	100	10	10	1
final value	−351.04	−358.48	−362.8	−367.38	−368.65	−368.83	−371.67	−378.16

this case the CMA-ES clearly outperforms the NAC while still being robust with respect to the initialisation of the policy parameters and the choice of the initial step size, see figure 5 and tables 3 and 4. NAC performs at best on par with stochastic search, for the more difficult policy initialisation P_{100} it is even worse.

5 Conclusion

The covariance matrix adaptation evolution strategy (CMA-ES) applied to reinforcement learning (RL) is conceptually similar to policy gradient methods

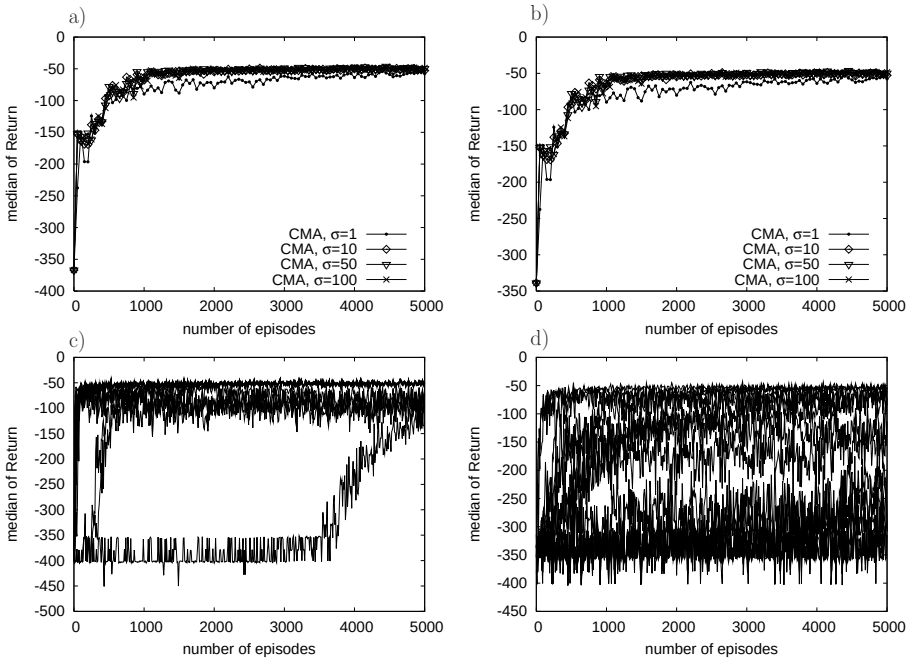

Fig. 4. Robustness against changes in the respective parameters on the mountain car task without noise. To avoid overcrowding plots only the worst-case examples are shown here: a) CMA-ES for initial policy P_0 and initial environment state S_{random} without noise and $\sigma \in [1, 10, 50, 100]$. b) CMA-ES for initial policy P_{100} and initial environment state S_{random} without noise and $\sigma \in [1, 10, 50, 100]$. c) NAC for initial policy P_0 and initial environment state S_{random} without noise, ordered from top to bottom by their final value as given in table 1, d) NAC for initial policy P_{100} and initial environment state S_{random} without noise, ordered from top to bottom by their final value as given in table 2.

with variable metric such as the natural actor critic (NAC) algorithm. However, we argue that the CMA-ES is much more robust w.r.t. the choice of hyperparameters, policy initialisation, and especially noise. On the other hand, given appropriate hyperparameters, the NAC can outperform the CMA-ES in terms of learning speed if initialised close to a desired policy. The experiments in this paper on the noisy mountain car problem and our previous results on the pole balancing benchmark support these conjectures. Across the different scenarios, the CMA-ES proved to be a highly efficient direct RL algorithm. The reasons for the robustness of the CMA-ES are the powerful adaptation mechanisms for the search distribution and the rank-based evaluation of policies.

In future work we will extend the experiments to different and more complex benchmark tasks and to other direct policy search methods.

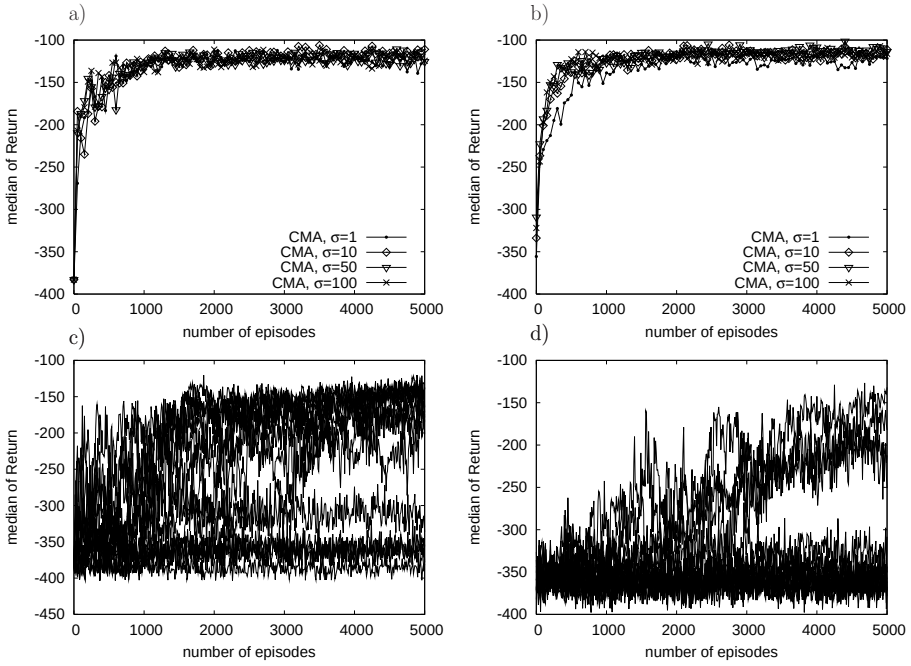

Fig. 5. Robustness against changes in the respective parameters on the mountain car task with noise. Again in order to avoid overcrowding plots only the worst-case examples are shown: a) CMA-ES for initial policy P_0 and initial environment state S_{random} with noise and $\sigma \in [1, 10, 50, 100]$. b) CMA-ES for initial policy P_{100} and initial environment state S_{random} with noise and $\sigma \in [1, 10, 50, 100]$. c) NAC for initial policy P_0 and initial environment state S_{random} with noise, ordered from top to bottom by their final value as given in table 3, d) NAC for initial policy P_{100} and initial environment state S_{random} with noise, ordered from top to bottom by their final value as given in table 4

Acknowledgement. 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".

References

1. Heidrich-Meisner, V., Igel, C.: Similarities and differences between policy gradient methods and evolution strategies. In Verleysen, M., ed.: 16th European Symposium on Artificial Neural Networks (ESANN), Evere, Belgium: d-side (2008) 149–154
2. Peters, J., Vijayakumar, S., Schaal, S.: Reinforcement learning for humanoid robotics. In: Proc. 3rd IEEE-RAS Int'l Conf. on Humanoid Robots. (2003) 29–30
3. Riedmiller, M., Peters, J., Schaal, S.: Evaluation of policy gradient methods and variants on the cart-pole benchmark. In: Proc. 2007 IEEE Internatinal Symposium

on Approximate Dynamic Programming and Reinforcement Learning (ADPRL 2007). (2007) 254–261
4. Peters, J., Schaal, S.: Applying the episodic natural actor-critic architecture to motor primitive learning. In: Proc. 15th European Symposium on Artificial Neural Networks (ESANN 2007), Evere, Belgien: d-side publications (2007) 1–6
5. Peters, J., Schaal, S.: Natural actor-critic. Neurocomputing **71**(7-9) (2008) 1180–1190
6. Hansen, N.: The CMA evolution strategy: A comparing review. In: Towards a new evolutionary computation. Advances on estimation of distribution algorithms. Springer-Verlag (2006) 75–102
7. Beyer, H.G.: Evolution strategies. Scholarpedia **2**(8) (2007) 1965
8. Igel, C.: Neuroevolution for reinforcement learning using evolution strategies. In: Congress on Evolutionary Computation (CEC 2003). Volume 4., IEEE Press (2003) 2588–2595
9. Pellecchia, A., Igel, C., Edelbrunner, J., Schöner, G.: Making driver modeling attractive. IEEE Intelligent Systems **20**(2) (2005) 8–12
10. Gomez, F., Schmidhuber, J., Miikkulainen, R.: Efficient non-linear control through neuroevolution. In: Proc. European Conference on Machine Learning (ECML 2006). Volume 4212 of LNCS., Springer-Verlag (2006) 654–662
11. Siebel, N.T., Sommer, G.: Evolutionary reinforcement learning of artificial neural networks. International Journal of Hybrid Intelligent Systems **4**(3) (2007) 171–183
12. Kassahun, Y., Sommer, G.: Efficient reinforcement learning through evolutionary acquisition of neural topologies. In Verleysen, M., ed.: 13th European Symposium on Artificial Neural Networks, d-side (2005) 259–266
13. Wierstra, D., Schaul, T., Peters, J., Schmidhuber, J.: Natural evolution strategies. In: IEEE World Congress on Computational Intelligence (WCCI 2008), IEEE Press (2008) Accepted.
14. Sutton, R., Barto, A.: Reinforcement Learning: An Introduction. MIT Press (1998)
15. Kaelbling, L., Littman, M., Cassandra, A.: Planning and acting in partially observable stochastic domains. Artificial Intelligence **101**(1-2) (1998) 99–134
16. Sutton, R., McAllester, D., Singh, S., Mansour, Y.: Policy gradient methods for reinforcement learning with function approximation. In: Advances in Neural Information Processing Systems. Volume 12. (2000) 1057–1063
17. Rechenberg, I.: Evolutionsstrategie: Optimierung Technischer Systeme nach Prinzipien der Biologischen Evolution. Frommann-Holzboog (1973)
18. Schwefel, H.P.: Evolution and Optimum Seeking. Sixth-Generation Computer Technology Series. John Wiley & Sons (1995)
19. Beyer, H.G., Schwefel, H.P.: Evolution strategies: A comprehensive introduction. Natural Computing **1**(1) (2002) 3–52
20. Kern, S., Müller, S., Hansen, N., Büche, D., Ocenasek, J., Koumoutsakos, P.: Learning probability distributions in continuous evolutionary algorithms – A comparative review. Natural Computing **3** (2004) 77–112
21. Hansen, N., Müller, S., Koumoutsakos, P.: Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). Evolutionary Computation **11**(1) (2003) 1–18
22. Hansen, N., Ostermeier, A.: Completely derandomized self-adaptation in evolution strategies. Evolutionary Computation **9**(2) (2001) 159–195
23. Hansen, N., Niederberger, A.S.P., Guzzella, L., Koumoutsakos, P.: A method for handling uncertainty in evolutionary optimization with an application to feedback control of combustion. IEEE Transactions on Evolutionary Computation (2008) In press.