Hoeffding and Bernstein Races for Selecting Policies in Evolutionary Direct Policy Search

Verena Heidrich-Meisner VERENA.HEIDRICH-MEISNER@NEUROINFORMATIK.RUB.DE
Christian Igel CHRISTIAN.IGEL@NEUROINFORMATIK.RUB.DE
Institut für Neuroinformatik, Ruhr-Universität Bochum, 44780 Bochum, Germany

Abstract

Uncertainty arises in reinforcement learning from various sources, and therefore it is necessary to consider statistics based on several roll-outs for evaluating behavioral policies. We add an adaptive uncertainty handling based on Hoeffding and empirical Bernstein races to the CMA-ES, a variable metric evolution strategy proposed for direct policy search. The uncertainty handling adjusts individually the number of episodes considered for the evaluation of a policy. The performance estimation is kept just accurate enough for a sufficiently good ranking of candidate policies, which is in turn sufficient for the CMA-ES to find better solutions. This increases the learning speed as well as the robustness of the algorithm.

1. Introduction

Dealing with uncertainty is one of the major issues in reinforcement learning (RL). When solving (partially observable) Markov decision processes solely based on observations and interactions with the environment, uncertainty and randomness arise from several sources. The initial state usually varies, state-transitions and reward signals can be stochastic, and the state observations may be noisy. We consider RL methods that search in a parametrized policy space, in which the search direction can be determined using estimates of the performance of behavioral policies or estimates of performance gradients. Uncertainty and randomness require that these estimates are based on a sample of several episodes (roll-outs). The sample size is a crucial parameter. If too few episodes are considered, the estimates are not reliable enough to allow for learning. If too many episodes are considered, the learning process gets too slow. Unfortunately, it is usually not possible to determine an appropriate sample size for a given problem a priori (in practice we just make it "large enough"). The optimal number may vary in the course of learning and between candidate policies (e.g., really bad behaviors can be detected after just a few roll-outs).

We employ the covariance matrix evolution strategy (CMA-ES, Hansen et al., 2003; Suttorp et al., 2009) for direct policy search, which gives striking results on RL benchmark problems (Gomez et al., 2008; Heidrich-Meisner and Igel 2008). The CMA-ES adapts the policy as well as parameters of its own search strategy (such as a variable metric) based on ranking policies. This is already much less susceptible to noise than estimating absolute performances or performance gradients (Heidrich-Meisner & Igel, 2008). Still, the ranking must be sufficiently accurate to evolve better policies, and the accuracy of the ranking depends on the degree of uncertainty as well as on the number of roll-outs considered per performance estimation of each candidate solution. We propose to augment the CMA-ES for RL with an adaptive uncertainty handling scheme based on Hoeffding or empirical Bernstein races (Maron & Moore, 1994; Maron & Moore, 1997; Audibert et al., 2007; Mnih et al., 2008), which dynamically adapts the number of episodes for evaluating a policy such that the ranking of new candidate solutions is just reliable enough to drive the learning process. All individuals participate in a selection race, in which their performances are sampled. A policy remains in the race as long as one cannot be sure with an a priori fixed probability whether the policy is promising or not, i.e., as long as the confidence interval for the estimated performance based on Hoeffding or empirical Bernstein bounds does not clearly distinguish it from the other candidates. The selection race is finished when a complete and accurate ranking is ob-

Appearing in *Proceedings of the 26th International Conference on Machine Learning*, Montreal, Canada, 2009.

tained.

In the next section, we describe the CMA-ES for direct policy search. In section 3 we introduce our new uncertainty handling scheme based on racing algorithms. Then we present some empirical results before we end with our conclusions.

2. Evolutionary direct policy search

Evolution strategies (ESs) are random search methods, which iteratively sample a set of candidate solutions from a probability distribution over the search space (here a parametrized space of policies), evaluate these potential solutions, and construct a new probability distribution over the search space based on the gathered information (Beyer, 2007). In ESs, this search distribution is parametrized 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.

Arguably the most elaborate ES for real-valued optimization is the covariance matrix adaptation ES (CMA-ES, Hansen et al., 2003; Suttorp et al., 2009), in which the main variation operator is additive Gaussian perturbation. The CMA-ES is a variable metric algorithm adapting shape and strength of its search distribution, and it is regarded as one of the most efficient evolutionary algorithms for real-valued optimization (Beyer, 2007). The CMA-ES is robust in the sense that it does not rely on tweaking of hyperparameters, see below. In a recent study, the CMA-ES (without rank-μ update, i.e., an outdated version) was compared to 8–12 (depending on the task) other RL algorithms including value-function and policy gradient approaches (Gomez et al., 2008). On the four test problems where the CMA-ES was considered, it ranked first, second (twice), and third. For further examples of successful applications of the CMA-ES for RL (Pellecchia et al., 2005; Siebel & Sommer, 2007) and additional comparisons on RL benchmarks (Heidrich-Meisner and Igel 2008; 2009) we refer to the literature.

In each generation k of the CMA-ES, which is shown in Algorithm 1, the lth offspring $\boldsymbol{x}_l^{(k+1)} \in \mathbb{R}^n$, $l \in \{1,\ldots,\lambda\}$, is generated by additive multi-variate *Gaussian mutation* and *weighted global intermediate recombination*, i.e., $\boldsymbol{x}_l^{(k+1)} \leftarrow \boldsymbol{m}^{(k)} + \sigma^{(k)}\boldsymbol{z}_l^{(k)}$ with mutation $\sigma^{(k)}\boldsymbol{z}_l^{(k)} \sim \sigma^{(k)}\mathcal{N}(\boldsymbol{0}, \boldsymbol{C}^{(k)})$ and recombination $\boldsymbol{m}^{(k)} \leftarrow \sum_{l=1}^{\mu} w_l \boldsymbol{x}_{l:\lambda}^{(k)}$. Here $\boldsymbol{x}_{l:\lambda}^{(k)}$ denotes the lth best individual of the λ offspring. Ranking the λ candidates (policies) requires their evaluation, which is discussed in detail in section 3. Considering the best μ of the offspring in the recombination implements non-elitist, rank-based selection. For the equal recombination used here A common choice for the recombination weights is $w_l \propto \ln(\mu+1) - \ln(l)$, $\|\boldsymbol{w}\|_1 = 1, \boldsymbol{w} \in \mathbb{R}^\mu$.

Algorithm 1: rank-μ CMA-ES

initialize $\boldsymbol{m}^{(0)} = \boldsymbol{x}_{\text{init}}$, $\sigma^{(0)}$, evolution paths $\boldsymbol{p}_\sigma^{(0)} = \boldsymbol{p}_c^{(0)} = \boldsymbol{0}$ and covariance matrix $\boldsymbol{C}^{(0)} = \boldsymbol{I}$ (unity matrix), $t_{\text{limit}}^{(0)} = 3$
// k counts number of generations
for $k = 0, \ldots$ **do**
  // create new offspring
  for $l = 1, \ldots, \lambda$ **do** $x_l^{(k+1)} \sim \mathrm{N}(\boldsymbol{m}^{(k)}, {\sigma^{(k)}}^2 \boldsymbol{C}^{(k)})$
  // evaluate new offspring, see section 3
  // $\hat{X}_l^{(k+1)}$ reflects quality of $\boldsymbol{x}_l^{(k+1)}$, $l = 1, \ldots, \lambda$
  $(\{\hat{X}_1^{(k+1)}, \ldots, \hat{X}_\mu^{(k+1)}\}, t_{\text{limit}}^{(k+1)}) \leftarrow$
  `selectRace`
  $(\{\boldsymbol{x}_1^{(k+1)}, \ldots, \boldsymbol{x}_\lambda^{(k+1)}\}, \mu, a, b, t_{\text{limit}}^{(k)}, \delta)$
  // recombination and selection
  $\boldsymbol{m}^{(k+1)} \leftarrow \sum_{i=1}^{\mu} w_i \boldsymbol{x}_{i:\lambda}^{(k+1)}$
  // step size control
  $\boldsymbol{p}_\sigma^{(k+1)} \leftarrow (1-c_\sigma)\boldsymbol{p}_\sigma^{(k)} + \sqrt{c_\sigma(2-c_\sigma)\mu_{\text{eff}}}\boldsymbol{C}^{(k)^{-\frac{1}{2}}} \frac{\boldsymbol{m}^{(k+1)} - \boldsymbol{m}^{(k)}}{\sigma^{(k)}}$
  $\sigma^{(k+1)} \leftarrow \sigma^{(k)} \exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{\|\boldsymbol{p}_\sigma^{(k+1)}\|}{\mathbb{E}[\|\mathcal{N}(\boldsymbol{0},\boldsymbol{I})\|]} - 1\right)\right)$
  // covariance matrix update
  $\boldsymbol{p}_c^{(k+1)} \leftarrow (1-c_c)\boldsymbol{p}_c^{(k)} + \sqrt{c_c(2-c_c)\mu_{\text{eff}}}\frac{\boldsymbol{m}^{(k+1)} - \boldsymbol{m}^{(k)}}{\sigma^{(k)}}$
  $\boldsymbol{C}^{(k+1)} \leftarrow (1-c_{\text{cov}})\boldsymbol{C}^{(k)} + \frac{c_{\text{cov}}}{\mu_{\text{cov}}}\boldsymbol{p}_c^{(k+1)}\boldsymbol{p}_c^{(k+1)^T} + c_{\text{cov}}\left(1 - \frac{1}{\mu_{\text{cov}}}\right)\sum_{i=1}^{\mu} w_i \boldsymbol{z}_{i:\lambda}^{(k)} \boldsymbol{z}_{i:\lambda}^{(k)^T}$

The CMA-ES adapts both the n-dimensional covariance matrix $\boldsymbol{C}^{(k)}$ of the normal mutation distribution as well as the *global step size* $\sigma^{(k)} \in \mathbb{R}^+$. The covariance matrix update has two parts, the rank-1 update considering the change of the population mean over time and the rank-μ update considering the successful variations in the last generation. The rank-1 update is based on a low-pass filtered *evolution path* $\boldsymbol{p}^{(k)}$ of successful (i.e., selected) steps

$$\boldsymbol{p}_c^{(k+1)} \leftarrow (1-c_c)\,\boldsymbol{p}_c^{(k)} + \sqrt{c_c(2-c_c)\mu_{\text{eff}}}\,\frac{\boldsymbol{m}^{(k+1)} - \boldsymbol{m}^{(k)}}{\sigma^{(k)}}$$

and aims at changing $\boldsymbol{C}^{(k)}$ to make steps in the promising direction $\boldsymbol{p}^{(k+1)}$ more likely by morphing the covariance towards $\left[\boldsymbol{p}_c^{(k+1)}\right]\left[\boldsymbol{p}_c^{(k+1)}\right]^T$. The backward time horizon of the cumulation process is approximately c_c^{-1}, where $c_c = 4/(n+4)$ is roughly inversely linear in the dimension of the path vector. The *variance effective selection mass* $\mu_{\text{eff}} = \left(\sum_{l=1}^{\mu} w_l^2\right)^{-1}$ is a normalization constant. The rank-μ update aims

at making the single steps that were selected in the last iteration more likely by morphing $\boldsymbol{C}^{(k)}$ towards $\left[\boldsymbol{z}_{i:\lambda}^{(k)}\right]\left[\boldsymbol{z}_{i:\lambda}^{(k)}\right]^T$. Putting both updates together, we have

$$\boldsymbol{C}^{(k+1)} \leftarrow (1-c_{\text{cov}})\boldsymbol{C}^{(k)} + \frac{c_{\text{cov}}}{\mu_{\text{cov}}}\boldsymbol{p}_c^{(k+1)}\boldsymbol{p}_c^{(k+1)^T} + c_{\text{cov}}\left(1-\frac{1}{\mu_{\text{cov}}}\right)\sum_{i=1}^{\mu} w_i \boldsymbol{z}_{i:\lambda}^{(k)}\boldsymbol{z}_{i:\lambda}^{(k)^T} \quad .$$

The constants c_{cov} and μ_{cov} are fixed learning rates. The learning rate of the covariance matrix update $c_{\text{cov}} = \frac{2}{(n+\sqrt{2})^2}$ is roughly inversely proportional to the degrees of freedom of the covariance matrix. The parameter μ_{cov} mediates between the rank-μ update ($\mu_{\text{cov}} \to \infty$) and the rank-one update ($\mu_{\text{cov}} = 1$). The default value is $\mu_{\text{cov}} = \mu_{\text{eff}}$.

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:

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

where the (*conjugate*) evolution path is

$$\boldsymbol{p}_\sigma^{(k+1)} \leftarrow (1-c_\sigma)\,\boldsymbol{p}_\sigma^{(k)} + \sqrt{c_\sigma(2-c_\sigma)\mu_{\text{eff}}}\,\boldsymbol{C}^{(k)^{-\frac{1}{2}}}\frac{\boldsymbol{m}^{(k+1)} - \boldsymbol{m}^{(k)}}{\sigma^{(k)}} \quad .$$

Again, $c_\sigma = \frac{\mu_{\text{eff}}+2}{n+\mu_{\text{eff}}+3}$ is a fixed learning rate and $d_\sigma = 1 + 2\max\left(0, \sqrt{\frac{\mu_{\text{eff}}-1}{n+1}}\right) + c_\sigma$ is a damping factor. The matrix $\boldsymbol{C}^{-\frac{1}{2}}$ is defined as $\boldsymbol{B}\boldsymbol{D}^{-1}\boldsymbol{B}^T$, where $\boldsymbol{B}\boldsymbol{D}^2\boldsymbol{B}^T$ 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 $\mathcal{N}(\boldsymbol{0},\boldsymbol{C})$ is done by sampling $\boldsymbol{B}\boldsymbol{D}\mathcal{N}(\boldsymbol{0},\boldsymbol{I})$.

The values of the learning rates and the damping factor are well considered and have been validated by experiments on many basic test functions (Hansen et al., 2003). *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. If we fix $\boldsymbol{C}^{(0)} = \boldsymbol{I}$, the only hyperparameter to be chosen problem dependent is the initial global step size $\sigma^{(0)}$ (apart from the parameters controlling sample size and uncertainty handling, see section 3).

3. Selection races

Due to uncertainty and randomness, the performance estimates, which are required for selecting policies, should be based on samples of several episodes (roll-outs). Our goal is to ensure with a given confidence that the μ selected policies have indeed the best mean performances while using as few roll-outs as possible. To this end, we want to systematically control in each iteration of our direct policy search (i) the overall number of roll-outs, and (ii) the distribution of roll-outs among the candidate policies. This can be achieved by adopting *racing algorithms* (Maron & Moore, 1994; Maron & Moore, 1997) for selection.

We view the performance of the policy encoded by $\boldsymbol{x}_i$, $i \in \{1,\ldots,\lambda\}$, as a real-valued random variable X_i. The return (accumulated reward) $X_{i,t}$ of the tth episode corresponds to a realization of X_i. We assume that the returns are almost surely bounded with known bounds a and b, $\Pr(X_{i,t} \in [a,b]) = 1$, and define $R = |a-b|$. Further, we assume an a priori fixed confidence level $1-\delta$ and a maximum number of evaluations per individual t_{limit} (alternatively, we may fix a maximum number of evaluations per iteration). If we consider the empirical mean $\hat{X}_{i,t} = \frac{1}{t}\sum_{t'=1}^{t} X_{i,t'}$ of t realizations (i.e., a performance estimate based on t episodes), Hoeffding's inequality bounds the deviation of the empirical mean from the true expectation $\mathbb{E}[X_i]$. With probability of at least $1-\delta$ we have $\left|\hat{X}_{i,t} - \mathbb{E}[X_i]\right| \leq R\sqrt{\frac{\log\frac{2}{\delta}}{2t}}$. This rather loose bound may be improved by applying Bernstein's inequality instead, which requires the usually unknown standard deviation of X_i. However, the recently derived *empirical Bernstein bound* (Audibert et al., 2007; Mnih et al., 2008) depends only on the empirical standard deviation $\hat{\sigma}_{i,t}^2 = \frac{1}{t}\sum_{t'=1}^{t}(X_{i,t'} - \hat{X}_{i,t})^2$. With probability of at least $1-\delta$ it holds

$$\left|\hat{X}_{i,t} - \mathbb{E}[X_i]\right| \leq \hat{\sigma}_{i,t}\sqrt{\frac{2\log\frac{3}{\delta}}{t}} + \frac{3R\log\frac{3}{\delta}}{t} \quad .$$

The dependency on R decreases linearly with the sample size t and not just with its square root, but now the bound depends also on $\hat{\sigma}_{i,t}/\sqrt{t}$. If the standard deviation is small compared to R, this bound is tighter than the Hoeffding bound.

Racing algorithms are iterative methods trying to identify with a high probability the best among several options. In each iteration, a couple of options are sampled. In their standard formulation, which we adopt here, there is an upper bound on the number of iterations.

Procedure `selectRace`$(\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_\lambda\}, \mu, a, b, t_{\text{limit}}^{(k)}, \delta)$.

1 $\mathbb{S} = \emptyset$ // set of selected individuals
2 $\mathbb{D} = \emptyset$ // set of discarded individuals
3 $\mathbb{U} = \{\boldsymbol{x}_i \,|\, i = 1, \ldots, \lambda\}$ // set of undecided individuals
4 $t \leftarrow 1$
5 **forall** $\boldsymbol{x}_i \in \mathbb{U}$ **do**
6 $\quad X_{i,t} \leftarrow$ `performance`$(\boldsymbol{x}_i)$ // evaluate individual
7 $\quad \text{LB}_i \leftarrow a$, $\text{UB}_i \leftarrow b$ // init lower and upper bounds
8 $u_t = |\mathbb{U}|$
9 **while** $t < t_{\text{limit}}^{(k)} \wedge |\mathbb{S}| < \mu$ **do**
10 $\quad t \leftarrow t + 1$
11 $\quad u_t = |\mathbb{U}|$ // needed for confidence intervals
$\quad$ // reevaluate undecided individuals
12 $\quad$ **forall** $\boldsymbol{x}_i \in \mathbb{U}$ **do**
13 $\quad\quad X_{i,t} \leftarrow$ `performance`$(\boldsymbol{x}_i)$
14 $\quad\quad \hat{X}_i \leftarrow \frac{1}{t}\sum_{t'=1}^{t} X_{i,t'}$ // recompute mean
15 $\quad\quad$ compute new confidence interval $\left[\hat{X}_{i,t} - c_{i,t}, \hat{X}_i + c_{i,t}\right]$
16 $\quad\quad$ using Hoeffding or empirical Bernstein bound
$\quad\quad$ // update lower and upper bounds
17 $\quad\quad \text{LB}_i \leftarrow \max\left\{\text{LB}_i, \hat{X}_i - c_{i,t}\right\}$
18 $\quad\quad \text{UB}_i \leftarrow \min\left\{\text{UB}_i, \hat{X}_i + c_{i,t}\right\}$
19 $\quad$ **forall** $\boldsymbol{x}_i \in \mathbb{U}$ **do**
20 $\quad\quad$ **if** $\left|\left\{\boldsymbol{x}_j \in \mathbb{U} \,\middle|\, LB_i > UB_j\right\}\right| \geq \lambda - \mu - |\mathbb{D}|$ **then**
$\quad\quad\quad$ // probably among the best μ
21 $\quad\quad\quad \mathbb{S} \leftarrow \mathbb{S} \cup \{\boldsymbol{x}_i\}$ // select
22 $\quad\quad\quad \mathbb{U} \leftarrow \mathbb{U} \setminus \{\boldsymbol{x}_i\}$
23 $\quad\quad$ **if** $\left|\left\{\boldsymbol{x}_j \in \mathbb{U} \,\middle|\, UB_i < LB_j\right\}\right| \geq \mu - |\mathbb{S}|$ **then**
$\quad\quad\quad$ // probably not among the best μ
24 $\quad\quad\quad \mathbb{D} \leftarrow \mathbb{D} \cup \{\boldsymbol{x}_i\}$ // discard
25 $\quad\quad\quad \mathbb{U} \leftarrow \mathbb{U} \setminus \{\boldsymbol{x}_i\}$
$\quad$ // adapt $t_{\text{limit}}^{(k+1)}$ depending on whether $t_{\text{limit}}^{(k)}$ was large enough for selection with confidence $1-\delta$ or not
26 **if** $|\mathbb{S}| = \mu$ **then** $t_{\text{limit}}^{(k+1)} = \max\{3, \frac{1}{\alpha} t_{\text{limit}}^{(k)}\}$
27 **else** $t_{\text{limit}}^{(k+1)} = \min\{\alpha t_{\text{limit}}^{(k)}, t_{\max}\}$
28 **return** $\{\hat{X}_1, \ldots, \hat{X}_\lambda\}$, $t_{\text{limit}}^{(k+1)}$

Our algorithm is described in the procedure `selectRace`. For the rank-based selection procedure used in the ES, we need to know the best μ from λ policies. Initially, all policies are evaluated (line 6) and then labelled *undecided* (line 3). In every following iteration or racing step, all policies tagged *undecided* are reevaluated (line 11). The estimated mean performance (line 14) and confidence interval (lines 17 and 18) are updated for each resampled policy (see below). If the lower bound of a policy is better than the upper bounds of at least $\lambda - \mu$ other candidates, it is selected (lines 20–22). Figure 1 illustrates this idea. If the upper bound of a policy is worse than the lower bounds of at least μ other candidates, it is discarded (lines 23–25). If μ individuals are selected or the number of iterations exceeds t_{limit}, the race is finished.

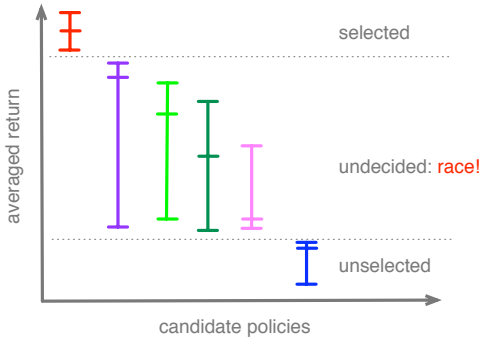

Figure 1. Example of a selection race for $\mu = 2$ and $\lambda = 6$.

We compute the confidence intervals using the Hoeffding or empirical Bernstein bound. If a confidence interval is recomputed, the highest lower and lowest upper bounds determined so far are stored. If we want our final decision to be correct with probability of at least $1 - \delta$, *all* computed bounds must be valid with probability of at least $1 - \delta$. By the union bound, this holds if each single bound holds with probability of at least $1 - \delta/n_{\text{b}}$, where n_{b} is the maximum number of considered bounds. In our setting, n_{b} can be upper bounded by λt_{limit}, because we have at most t_{limit} iterations and at most λ evaluations per iteration. Thus, after t evaluations of policy $\boldsymbol{x}_i$ we get a confidence interval $\left[\hat{X}_{i,t} - c_{i,t}, \hat{X}_i + c_{i,t}\right]$ with

$$c_{i,t}^{\text{Hoeffding}} = R\sqrt{\frac{\log(2n_{\text{b}}) - \log\delta}{2t}}$$

using the Hoeffding and

$$c_{i,t}^{\text{Bernstein}} = \hat{\sigma}_{i,t}\sqrt{2\frac{\log(3n_{\text{b}}) - \log\delta}{t}} + 3R\frac{\log(3n) - \log\delta}{t}$$

using the empirical Bernstein bound. The approximation $n_{\text{b}} = \lambda t_{\text{limit}}$ is much too loose because it does not consider that already selected or discarded individuals are not reevaluated. Therefore we use in iteration t

$$n_{\text{b},t} = \sum_{k=1}^{t-1} u_k + (t_{\text{limit}} - t + 1)u_t \ ,$$

where u_t is the number of individuals labeled undecided in iteration t.

We combine this selection algorithm based on racing with the CMA-ES and call the resulting algorithms Hoeffding- and Bernstein-Race-CMA-ES depending on

the way we compute the confidence intervals. By construction, our selection algorithm has the following property:

Proposition 1 *Let $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_\lambda\}$ be a set of individuals with fitness values almost surely between a and b, $\mu < \lambda$, $t_{\text{limit}} \geq 3$, and $\delta \in]0, 1]$. If the above procedure* `selectRace` *selects a set $\mathbb{S}$ of policies, then with probability of at least $1-\delta$ these elements belong to the best μ out of $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_\lambda\}$ in terms of mean performance.*

If after t_{limit} iterations less than μ individuals were selected, the – according to their estimated mean – best of the not yet discarded policies are considered in the CMA-ES. If this happens t_{limit} was too small, and therefore the maximum number of iterations is increased by a factor of $\alpha > 1$ in the next generation upper bounded by a threshold $t_{\max}$ (line 27). If t_{limit} iterations were sufficient, t_{limit} is reduced by a factor of α^{-1} (line 26), which has a positive effect on the bounds and therefore can speed up future races. Thus, the selection procedure adapts the overall budget of policy evaluations and distributes the available evaluations among the candidate policies.

Related work. Stagge (1998) and Schmidt et al. (2006) propose methods for uncertainty handling in rank-based evolution strategies for noisy function optimization. However, both heuristics assume a particular distribution of the performance samples. Heidrich-Meisner and Igel (2009) use the uncertainty handling CMA-ES (UH-CMA-ES) developed by Hansen et al. (2009) for reliably ranking policies in RL. However, this heuristic only adjusts globally in each generation the number of roll-outs considered for evaluation. Racing algorithms have been proposed in the domain of evolutionary computation for the empirical evaluation of different evolutionary algorithms (e.g., different external strategy parameter configurations) (Birattari et al., 2002; Yuan & Gallagher, 2004), but to the best of our knowledge not yet for selection.

4. Experiments

Benchmark problems. We consider RL benchmarks taken from the literature. The mountain car task serves as a minimal example. Here an underpowered car has to be driven out a valley to the goal state at the hilltop (Sutton & Barto, 1998). The state $\boldsymbol{s} = [x, \dot{x}]^T$ of the system is given by the position $x \in [-1.2, 0.6]$ of the car and by its current velocity $\dot{x} \in [-0.07, 0.07]$. Actions are discrete forces applied to the car $a \in \{-a_{\max}, 0, a_{\max}\}$. In every time step a reward of -1 is given to the agent. The initial state is uniformly drawn from $[-1.2, 0.6] \times [-0.07, 0.07]$. In a second scenario we add Gaussian noise $\mathcal{N}(0, 0.01)$ to the state observation (Heidrich-Meisner & Igel, 2008), making the underlying Markov decision process partially observable. A trial is *successful* if the final policy allows the car to reach the hilltop in less than 100 time steps on average for the fully observable task and in less than 120 time steps on average for the partially observable task.

As a higher dimensional and more challenging task we consider the swimmer problem (Coulom, 2002). The swimmer consists of n_{c} compartments floating on a liquid surface. The swimmer is supposed to move its center of gravity as fast as possible in a predefined direction. The state description $\boldsymbol{s} = [A_0, \zeta_1, \ldots, \zeta_{n_{\text{c}}}, \dot{A}_0, \dot{\zeta}_1, \ldots, \dot{\zeta}_{n_{\text{c}}}]^T$ includes the position of the end point of the first compartment A_0 (marking the "head" of the swimmer), the angle of the ith part ($i = 1, \ldots, n_{\text{c}}$) with respect to the x-axis, the corresponding velocity of the "head" $\dot{A}_0$, and the angular velocities for each part $\dot{\zeta}_1, \ldots, \dot{\zeta}_{n_{\text{c}}}$. Actions $\boldsymbol{a} = [a_1, \ldots, a_{n_{\text{c}}-1}]^T$ are torques applied between body parts. The reward given to the swimmer is the velocity component of the swimmer's center of gravity parallel to the x-axis. We considered two swimmers with $n_{\text{c}} \in \{3, 4\}$. The swimmer's initial position is set to $\mathbf{0}$ and the initial angular velocities are each drawn uniformly from $[0, \pi]$. A swimmer trial is called successful if the average velocity of the swimmer's center of gravity is larger than $\frac{3n_{\text{c}}}{40}\frac{\text{m}}{\text{s}}$, i.e., the swimmer covers a distance of at least one and a half of its length in the desired direction in the simulated time span of 20 s.

Baseline algorithms. In order to judge the performance of the CMA-ES and the two Race-CMA-ESs for RL, we consider two alternative methods for comparison. First, we use simple random search (random weight guessing, RWG) as a baseline for evaluation. In every iteration new policy parameters are drawn uniformly from an interval $[-x_{\max}, x_{\max}]^n$, where n is the number of policy parameters, and the best solution is maintained. Second, we apply the efficient episodic natural actor-critic algorithm (NAC) according to Peters and Schaal (2008), a state-of-the-art policy gradient method. The NAC is, like the CMA-ES, a variable metric algorithm operating on a predefined policy class.

Experimental setup. In all four benchmark problems uncertainty arises from random start states. In the second of the two mountain car tasks we additionally have to cope with noisy observations. We always consider the same type of linear policies in order to allow for a fair comparison. The linear poli-

cies examined here are typically used with the NAC (Peters & Schaal, 2008). More sophisticated choices of policy classes certainly improve the performance of the CMA-ES, which for instance works fine with non-linear neural networks (Pellecchia et al., 2005; Siebel & Sommer, 2007; Gomez et al., 2008). Thus all methods operate on the same policy class $\pi^{\text{deter}}_{\boldsymbol{x}}(\boldsymbol{s}) = \boldsymbol{x}^T\boldsymbol{s}$ with $\boldsymbol{s}, \boldsymbol{x} \in \mathbb{R}^n$. For learning, the NAC uses the stochastic policy $\pi^{\text{stoch}}_{\boldsymbol{x}}(\boldsymbol{s}, a) = \mathcal{N}(\pi^{\text{deter}}_{x}(\boldsymbol{s}), \sigma_{\text{NAC}})$, where the standard deviation σ_{NAC} is considered as an additional adaptive parameter of the policy gradient method. The NAC is evaluated on the corresponding deterministic policy. The policy parameters (except the exploration parameter σ_{NAC} for the NAC) are always initialized with zero. For the swimmer task the action consists of $n_{\text{c}} - 1$ continuous values and we apply an independent linear policy for each action component, thus the search space is $2(n_{\text{c}}^2 - 1)$-dimensional. For the independent evaluation of the algorithms (e.g., in the following plots), we determine the median of 50 roll-out for the mountain car tasks and of 10 roll-outs for the swimmer tasks. For the mountain car problems we test for the CMA-ES all combinations of initial global step size $\sigma^{(0)} \in \{0.1, 1, 5, 10, 15, 25, 50, 100\}$ and sample size $n_{\text{eval}} \in \{1, 10, 20, 30\}$, and for the NAC all combinations of initial exploration $\sigma_{\text{NAC}} \in \{0.1, 1, 5, 10, 15, 25, 50, 100\}$, learning rate $\alpha_{\text{NAC}} \in \{0.1, 0.01, 0.001, 0.0001\}$, and sample size $n_{\text{eval}} \in \{n+2, 2(n+2), 3(n+2)$ (the NAC needs a minimum of $n+2$ roll outs per policy update). Since the swimmer problem is the more computational demanding we restricted $\sigma^{(0)}$ and σ_{NAC} to $\{1, 10, 50\}$ for this task.

For the Hoeffding- and Bernstein-Race-CMA-ES we always test all combinations of confidence $\delta \in \{0.01, 0.05, 0.1\}$ and $\sigma^{(0)} \in \{1, 10, 50\}$. In each trial we initialize $t_{\text{limit}} = 3$ (at least three roll-outs are necessary to compute the empirical Bernstein bounds), and we use $\alpha = 1.5$ and $t_{\max} = 50$ as upper bound on t_{limit}.

Results. In Figs. 2 and 3 the performances of RWG, CMA-ES, NAC, and Bernstein- and Hoeffding-Race-CMA-ES are shown for the four test scenarios. For all methods the performance for the best respective parameter configuration is plotted. The race based selections clearly increased the learning speed. RWG, NAC, and CMA-ES performed best for large but not too large sample sizes n_{eval}. Among the methods without uncertainty handling the CMA-ES was more robust against uncertainty than NAC and RWG when looking at the best respective parameter configurations. In the fully observable mountain car problem the NAC performed exceptionally well because it benefits from a policy initialization close to an optimum (Heidrich-Meisner & Igel, 2008). However, in the partially observable mountain car problem the Race-CMA-ESs

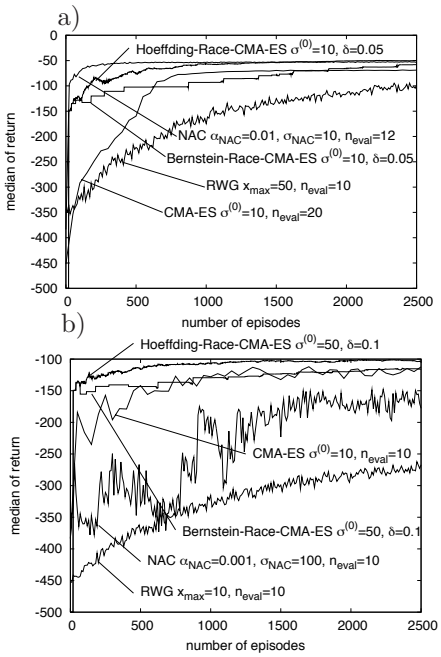

Figure 2. Median of performance over 500 trials for RWG, NAC, CMA-ES, Bernstein-Race-CMA-ES, and Hoeffding-Race-CMA-ES in the a) fully observable and b) partially observable mountain car task. The respective best parameter configuration is plotted.

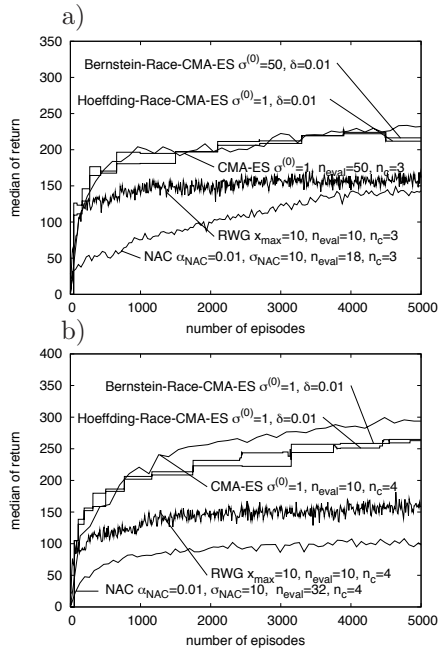

Figure 3. Median of performance over 20 trials for RWG, NAC, CMA-ES, Bernstein-Race-CMA-ES, and Hoeffding-Race-CMA-ES for swimmers with a) 3 and b) 4 segments. The respective best parameter configuration is plotted.

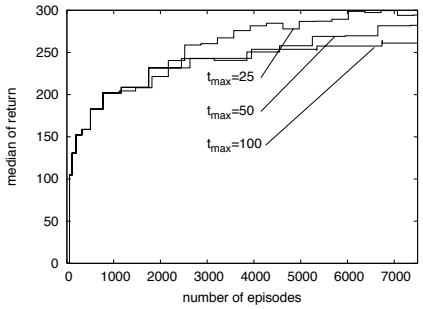

Figure 4. Median of the performance over 20 trials for the Bernstein-Race-CMA-ES with $\delta = 0.05$ and $\sigma^{(0)} = 1$ for different maximal race lengths $t_{\max} \in \{25, 50, 100\}$ in the swimmer task with $n_c = 4$.

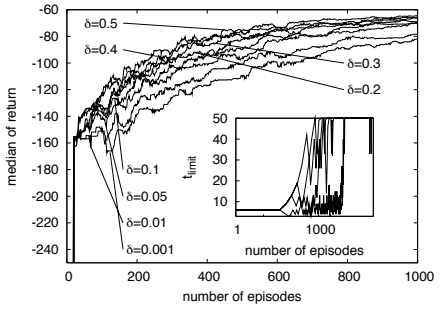

Figure 5. Median of performance over 500 trials for the Hoeffding-Race-CMA-ES with $\sigma^{(0)} = 50$ in the partially observable mountain car task for different confidence levels $\delta \in \{0.001, 0.01, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5\}$. The race length t_{limit} selected by the Hoeffding-Race-CMA-ES with $\lambda = 5$, $t_{\max} = 50$ and $\delta = 0.05$ for four selected trials is plotted in the inset.

outperformed all other methods. For both mountain car tasks the Hoeffding-Race-CMA-ES performed better than the Bernstein-Race-CMA-ES. Initializing the start state of the mountain car randomly led to a large standard deviation $\hat{\sigma}_{i,t}$ compared to the range R and therefore the Hoeffding bound was tighter than the empirical Bernstein bound. In the swimmer tasks the empirical Bernstein bound was tighter than the Hoeffding bound, and the Bernstein-Race-CMA-ES outperformed the Hoeffding-Race-CMA-ES. The performance of both Race-CMA-ESs depended on the upper bound $t_{\max}$ of the race length t_{limit} as shown in Fig. 4 for the Bernstein-Race-CMA-ES. Choosing $t_{\max}$ too large results in very costly policy updates and unnecessary slow learning. This can be seen in the swimmer task with $n_c = 4$, where both Race-CMA-ESs performed poorly even though they initially improve the learning speed. The inset in Fig. 5 shows the values of t_{limit} chosen by the Bernstein-Race-CMA-ES for single trials. At the beginning small sample sizes were used, and when the accuracy was no longer sufficient the sample size was increased. Finally for fine tuning the maximal allowed sample size of $t_{\max} = 50$ was used most of the time. As can be seen in Fig. 5, a lower confidence improved the performance in our experiments, because in the initial learning phase, which is crucial for the overall performance in simple mountain car task, a high confidence is not needed.

But the Race-CMA-ESs not only improved the learning speed. They also proved to be very robust against changes in their hyperparameter values. For the mountain car task both the Bernstein- and the Hoeffding-Race-CMA-ES were successful in more than 80% of the 500 trials for 9 out of 9 tested parameter configurations, while the standard CMA-ES was successful in more than 80% of the trials in only 18 of 32 cases, the NAC in 60 of 96 cases and RWG in 7 of 28 cases. In the partially observable mountain car task, the Bernstein- and the Hoeffding-Race-CMA-ES were again successful in more than 80% of the trials for 9 out of 9 tested parameter configurations. CMA-ES, NAC, and RWG achieved a success level of 80% in 3 of 3, 6 of 12, and 0 of 28 cases, respectively. (In this case only the most effective fixed sample size was tested for the CMA-ES and the NAC.) For swimmers with 3 compartments the Race-CMA-ESs were also robust. The Bernstein-Race-CMA-ES was successful in more than 50% of all trials for 6 of 9 and the Hoeffding-Race-CMA-ES was successful in more than 50% of all trials for 5 of 9 and parameter configurations. For swimmers with $n_c = 4$ both Race-CMA-ESs were not successful because presumably the value of $t_{\max}$ was too large. The CMA-ES achieved successes in more than 50% of the trials in 11 of 12 and 5 of 12 cases for $n_c \in \{3, 4\}$, while the NAC and RWG reached this success level in none of the configurations. Thus, uncertainty handling through selection races remarkably sped up learning and improved at the same time the robustness compared to the CMA-ES without uncertainty handing, which is already much more robust than the policy gradient method.

5. Discussion and Conclusion

Evolution strategies (ESs) are powerful direct policy search methods. One of their main advantages is their ability to cope with uncertainty and noise. Still, random elements in the environment require gathering statistics over several episodes for the evaluation of candidate policies. We added a new adaptive uncertainty handling to evolutionary reinforcement learning. It adjusts the number of roll-outs per evaluation of a policy such that the signal to noise ratio is just high enough for a sufficiently good ranking of candidate policies, which in turn suffices for the ES to find

better solutions. The uncertainty handling exploits the advantage of small sample sizes in the beginning and increases the sample size when a higher accuracy is necessary. It balances the trade-off between fast learning and sufficient accuracy. This significantly increases both learning speed and robustness.

The statistically sound uncertainty handling scheme is independent of the CMA-ES and could be combined with other RL approaches (e.g., for evolutionary online RL, Whiteson & Stone, 2006). Moreover, it is not limited to RL and may also be adopted for general noisy function approximation tasks.

References

Audibert, J.-Y., Munos, R., & Szepesvári, C. (2007). Tuning bandit algorithms in stochastic environments. *18th International Conference on Algorithmic Learning Theory (ALT)* (pp. 150–165). Springer-Verlag.

Beyer, H.-G. (2007). Evolution strategies. *Scholarpedia*, *2*, 1965.

Birattari, M., Stutzle, T., Paquete, L., & Varrentrapp, K. (2002). A racing algorithm for configuring metaheuristics. *Genetic and Evolutionary Computation Conference (GECCO 2002)* (pp. 11–18). Morgan Kaufmann Publishers.

Coulom, R. (2002). Apprentissage par renforcement utilisant des reseaux de neurones, avec des applications au controle moteur. *These de doctorat, Institut National Polytechnique de Grenoble*.

Gomez, F., Schmidhuber, J., & Miikkulainen, R. (2008). Accelerated neural evolution through cooperatively coevolved synapses. *Journal of Machine Learning Research*, *9*, 937–965.

Hansen, N., Müller, S. D., & Koumoutsakos, P. (2003). Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). *Evolutionary Computation*, *11*, 1–18.

Hansen, N., Niederberger, A. S. P., Guzzella, L., & Koumoutsakos, P. (2009). A method for handling uncertainty in evolutionary optimization with an application to feedback control of combustion. *IEEE Transactions on Evolutionary Computation*, *13*, 180–197.

Heidrich-Meisner, V., & Igel, C. (2008). Variable metric reinforcement learning methods applied to the noisy mountain car problem. *European Workshop on Reinforcement Learning (EWRL 2008)* (pp. 136–150). Springer-Verlag.

Heidrich-Meisner, V., & Igel, C. (2009). Uncertainty handling CMA-ES for reinforcement learning. *Genetic and Evolutionary Computation Conference (GECCO 2009)*. ACM Press.

Maron, O., & Moore, A. W. (1994). Hoeffding races: Accelerating model selection search for classification and function approximation. *Advances in Neural Information Processing Systems* (pp. 59–66). Morgan Kaufmann Publishers.

Maron, O., & Moore, A. W. (1997). The racing algorithm: Model selection for lazy learners. *Artificial Intelligence Review*, *11*, 193–225.

Mnih, V., Szepesvári, C., & Audibert, J. (2008). Empirical Bernstein stopping. *Proceedings of the 25th International Conference on Machine Learning (ICML 2008)* (pp. 672–679).

Pellecchia, A., Igel, C., Edelbrunner, J., & Schöner, G. (2005). Making driver modeling attractive. *IEEE Intelligent Systems*, *20*, 8–12.

Peters, J., & Schaal, S. (2008). Natural actor-critic. *Neurocomputing*, *71*, 1180–1190.

Schmidt, C., Branke, J., & Chick, S. (2006). Integrating techniques from statistical ranking into evolutionary algorithms. *Applications of Evolutionary Computing* (pp. 752–763). Springer-Verlag.

Siebel, N. T., & Sommer, G. (2007). Evolutionary reinforcement learning of artificial neural networks. *International Journal of Hybrid Intelligent Systems*, *4*, 171–183.

Stagge, P. (1998). Averaging efficiently in the presence of noise. *Parallel Problem Solving from Nature (PPSN V)* (pp. 188–197). Springer-Verlag.

Sutton, R., & Barto, A. (1998). *Reinforcement learning: An introduction*. MIT Press.

Suttorp, T., Hansen, N., & Igel, C. (2009). Efficient covariance matrix update for variable metric evolution strategies. *Machine Learning*, *75*, 167–197.

Whiteson, S., & Stone, P. (2006). Evolutionary function approximation for reinforcement learning. *Journal of Machine Learning Research*, *7*, 877–917.

Yuan, B., & Gallagher, M. (2004). Statistical racing techniques for improved empirical evaluation of evolutionary algorithms. *Parallel Problem Solving from Nature (PPSN VIII)* (pp. 172–181). Springer-Verlag.