JMLR: Workshop and Conference Proceedings 80:1–16, 2017 ACML 2017

# A Covariance Matrix Adaptation Evolution Strategy for Direct Policy Search in Reproducing Kernel Hilbert Space

**Ngo Anh Vien** V.NGO@QUB.AC.UK
*EEECS, Queen's University Belfast, United Kingdom*

**Viet-Hung Dang** DANGVIETHUNGHA@GMAIL.COM
*Research and Development Center, Duy Tan University, Vietnam*

**TaeChoong Chung** TCCHUNG@KHU.AC.KR
*Department of Computer Engineering, Kyung Hee University, Korea*

**Editors:** Yung-Kyun Noh and Min-Ling Zhang

## Abstract

The covariance matrix adaptation evolution strategy (CMA-ES) is an efficient derivative-free optimization algorithm. It optimizes a black-box objective function over a well defined parameter space. In some problems, such parameter spaces are defined using function approximation in which feature functions are manually defined. Therefore, the performance of those techniques strongly depends on the quality of chosen features. Hence, enabling CMA-ES to optimize on a more complex and general function class of the objective has long been desired. Specifically, we consider modeling the input space for black-box optimization in reproducing kernel Hilbert spaces (RKHS). This modeling leads to a functional optimization problem whose domain is a function space that enables us to optimize in a very rich function class. In addition, we propose CMA-ES-RKHS, a generalized CMA-ES framework, that performs black-box functional optimization in RKHS. A search distribution, represented as a Gaussian process, is adapted by updating both its mean function and covariance operator. Adaptive representation of the mean function and the covariance operator is achieved by resorting to sparsification. CMA-ES-RKHS is evaluated on two simple functional optimization problems and two bench-mark reinforcement learning (RL) domains. For an application in RL, we model policies for MDPs in RKHS and transform a cumulative return objective as a functional of RKHS policies, which can be optimized via CMA-ES-RKHS. This formulation results in a black-box functional policy search framework.

**Keywords:** CMA-ES, functional optimization, policy search, kernel methods, RKHS

## 1. Introduction

The covariance matrix adaptation evolutionary strategy (CMA-ES) is a derivative-free method (Hansen et al., 2003) that is a practical optimizer for continuous optimization problems. It is a general optimization framework that possesses many appealing characteristics, e.g. derivative-free, covariant, off-the-shelf, scalable etc.. It is especially useful on problems that are non-convex, non-separable, ill-conditioned, multi-modal, and with noisy evaluations. CMA-ES belongs to Evolution Strategy (ES), hence it also works by operating major steps: recombination, mutation and selection. In the context of robotics, CMA-ES has been widely used in many tasks: biped locomotion (Wang et al., 2012), whole-body locomotion optimization (Ha and Liu, 2014, 2016), swimming (Tan et al., 2011), skill learning

via reinforcement learning (Heidrich-Meisner and Igel, 2009a,b; Stulp and Sigaud, 2012), inverse reinforcement learning (Rückert et al., 2013; Doerr et al., 2015), etc. Applying CMA-ES requires explicitly a finite-dimensional search space on which solution candidates live. In many domains, e.g. robotics, an optimization objective is often defined as a function of another parametric function. For instance, it might be an overall cost function depending on a robot controller, e.g. robot skill learning (Stulp and Sigaud, 2012), policy search (Heidrich-Meisner and Igel, 2009a), or a loss function in contexts of inverse optimal control (Rückert et al., 2013; Doerr et al., 2015), etc.. A robot controller is usually a parametric function of predefined feature maps, e.g. RBF features, neural networks. Though showing a lot of remarkable successes, such parametric approaches as well as their black-box solver, the parametric CMA-ES, could not model a very rich and flexible solution space, e.g. a complex behavior space.

In this work, we propose CMA-ES-RKHS that enables functional optimization over a non-parametric solution space. Specifically, we assume that the solution space is a reproducing kernel Hilbert space (RKHS). Each candidate is a function in RKHS. Modeling the solution space this way, CMA-ES-RKHS can not only inherit full characteristics from CMA-ES, but also enjoy other appealing properties. *Firstly*, CMA-ES-RKHS is able to optimize a functional objective whose domain is an RKHS. That means the solution space does not need to depend on any manual parametrization. *Secondly*, by modeling the solution space in RKHS, all update steps in CMA-ES-RKHS are handled analytically. We show that updated mean functionals, other intermediate terms, evolution path functionals or conjugate evolution path functionals are functions in the underlying RKHS. Moreover, the updated covariance is also an operator on the underlying RKHS. *Thirdly*, via sparsification in RKHS, a very complex search space can be represented compactly, however we can still achieve a solution of guaranteed quality.

By employing CMA-ES-RKHS, we propose a non-parametric direct policy search technique in which the policy space is an RKHS. As shown by (Bagnell and Schneider, 2003; Lever and Stafford, 2015; van Hoof et al., 2015; Vien et al., 2016; Tuyen et al., 2017), the policy in RKHS can be powerful and flexible in solving complex tasks. However their approaches use functional gradient ascent to update the policy functional that would have a problem with step-sizes or local optima. It means the optimized policy does not fully exploit the flexible power of modeling in RKHS, which consists of any complex policies, including the global optimal one. As a global optimization method, our policy search via CMA-ES-RKHS is expected to search for such globally optimal policies.

We demonstrate the new CMA-ES-RKHS and direct policy search via CMA-ES-RKHS in four experiments. The first two synthetic domains demonstrate the advantages of functional optimization in RKHS. The last two experiments are two RL domains, inverted pendulum and double-pendulum swing-up.

## 2. Background

We briefly present a background of CMA-ES, its application for direct policy search reinforcement learning, and the recently introduced policy search in RKHS.

### 2.1. Covariance Matrix Adaptation - Evolution Strategy

The Covariance Matrix Adaptation - Evolution Strategy (CMA-ES) is a global optimization method introduced by (Hansen et al., 2003). It works by forming a parametric distribution over the solution space, e.g. the space of policy parameter in policy search, or the space of parameters of the loss function in inverse optimal control, etc. It iteratively samples a population of solution candidates from a parametrized search distribution. These candidates are then evaluated by a black-box function. Tuples of candidate-evaluation make up a dataset in order for CMA-ES to update the search distribution, i.e. its mean and its covariance matrix. Specifically, a cost function $f(\theta)$ is parametrized by a parameter space $\theta \in \Re^n$, $f : \Re^n \mapsto \Re$. It is common that a CMA-ES algorithm maintains a multi-variate Gaussian distribution over the solution space as $\theta \sim \mathcal{N}(\theta; \mathbf{m}, \mathbf{C})$. At each iteration $k$, it generates the $k$th population of $\lambda$ candidates from the $k$th distribution as $\theta_i \sim \mathcal{N}(\theta; \mathbf{m}_k, \mathbf{C}_k)$, $i = 1, \cdots, \lambda$. Then, the candidates are sorted ascendingly according to their evaluations $f(\theta_i)$. Only the first $\mu$ best candidates are selected for use in updates of $\mathbf{m}_k$ and $\mathbf{C}_k$. Another parameter is the global step-size $\sigma \in \Re$ that controls the convergence rate of the covariance matrix update. The parameter $\sigma$ is defined as a global standard deviation. Hence, a full set of parameters in CMA-ES is $\{\mathbf{m}, \mathbf{C}, \sigma\}$.

In Algorithm 1, we give a full summary of the CMA-ES algorithm. The updated mean is a weighted sum of the best $\mu$ candidates as in step 7, in which the weights $w_i$ are set to $1/\mu$ or to a better choice $\log(\mu/2) - \log(i)$. The notation $\mathbf{y}_{i:\lambda}$ means the best candidate out of $\mathbf{y}_i, \ldots, \mathbf{y}_\lambda$. The covariance matrix update in step 13 consists of three parts: old information, rank-1 update which computes the change of the mean over time as encoded in the *evolution path* $\mathbf{p}_c$, and rank-$\mu$ update which takes into account the *good* variations in the last population. This step-size control update constrains the expected changes of the distribution. Thus, this update in step 10 is based on the *conjugate evolution path* $\mathbf{p}_\sigma$. It aims to accelerate convergence to an optimum, and meanwhile prevents premature convergence. The other parameters: $\mu_w$ is the variance effective selection mass, $c_1, c_c, c_\sigma$ are learning rates, and $d_\sigma$ is a damping factor for $\sigma$. The setting of these parameters is well studied and discussed in-depth in (Hansen, 2016).

The updates of CMA-ES can alternatively be derived using the information-geometric concept of a natural gradient as shown in (Hansen and Auger, 2014), which shares the same insight with the *natural evolution strategies* (NES) (Wierstra et al., 2014).

There are less efficient techniques that can also adapt the covariance matrix: *estimation of distribution algorithms* (EDA) and *the cross-entropy method* (CEM). The major difference from CMA-ES is at the choice of the reference mean value. EDA and CEM estimate the variance within the current population, $\mathbf{y}_{1:\lambda}$, instead of exploiting old information as encoded in previous $\mathbf{C}$ and $\mathbf{p}_c$. Specifically, for the Gaussian search distribution, analytic updates at iteration $k$ for EDA and CEM (Rubinstein and Kroese, 2013; Rubinstein, 1999)

**Algorithm 1** The CMA-ES algorithm

1: Initialize $\mathbf{m} \in \Re^n$, $\sigma \in \Re^+$, $\lambda, \mu$
2: Initialize $\mathbf{C} = \mathbf{I}, \mathbf{p}_c = \mathbf{0}, \mathbf{p}_\sigma = \mathbf{0}$
3: **while** not terminate **do**
4: Sampling: $\theta_i = \mathbf{m} + \sigma \mathbf{y}_i$, $\mathbf{y}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{C}), i = 1, \dots, \lambda$
5: Evaluating: $f(\theta_i), i = 1, \dots, \lambda$
6: *// mean update*
7: $\mathbf{m} \leftarrow \mathbf{m} + \sigma \bar{\mathbf{y}}$, where $\bar{\mathbf{y}} = \sum_1^\mu w_i \mathbf{y}_{i:\lambda}$
8: *// step-size control update*
9: $\mathbf{p}_\sigma \leftarrow (1 - c_\sigma)\mathbf{p}_\sigma + \sqrt{c_\sigma(2 - c_\sigma)\mu_w}\mathbf{C}^{-\frac{1}{2}}\bar{\mathbf{y}}$
10: $\sigma \leftarrow \sigma \exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{\|\mathbf{p}_\sigma\|}{\mathbb{E}\|\mathcal{N}(\mathbf{0},\mathbf{I})\|} - 1\right)\right)$
11: *// covariance matrix update*
12: $\mathbf{p}_c \leftarrow (1 - c_c)\mathbf{p}_c + \sqrt{c_c(2 - c_c)\mu_w}\bar{\mathbf{y}}$
13: $\mathbf{C} \leftarrow (1 - c_1 - c_\mu)\mathbf{C} + c_1 \mathbf{p}_c \mathbf{p}_c^\top + c_\mu \sum_1^\mu w_i \mathbf{y}_{i:\lambda}\mathbf{y}_{i:\lambda}^\top$
14: **end while**

change steps 9 and 13 in Algorithm 2 to

$$\mathbf{m}^{(k)} = \frac{1}{\mu}\sum_{i=1}^{\mu}\theta_i$$

$$\mathbf{C}_{\text{EDA}}^{(k)} = \frac{1}{\mu}\sum_{i=1}^{\mu}\left(\theta_i - \mathbf{m}^{(k)}\right)\left(\theta_i - \mathbf{m}^{(k)}\right)^\top$$

$$\mathbf{C}_{\text{CEM}}^{(k)} = \frac{\mu}{\mu - 1}\mathbf{C}_{\text{EDA}}^{(k)}$$

The difference between EDA and CEM is: EDA updates the empirical covariance matrix, meanwhile CEM updates the *unbiased* empirical covariance matrix

### 2.2. Direct policy search in RL

RL algorithms optimize a policy that maps states of the environment to the actions of an agent. The policy can be optimized using policy search techniques such as policy gradient (Williams, 1992), natural actor-critic (Peters and Schaal, 2008), or black-box search such as CEM (Mannor et al., 2003) and CMA-ES (Heidrich-Meisner and Igel, 2009a), etc. The first two techniques are local methods that can only look for locally optimal policies (more discussions over their differences can be found in (Heidrich-Meisner and Igel, 2008)). We first describe an application of CMA-ES for policy search as introduced in (Heidrich-Meisner and Igel, 2009a), then give a brief introduction of one policy gradient technique that models policies in reproducing kernel Hilbert space (Lever and Stafford, 2015).

#### 2.2.1. Direct policy search via CMA-ES

Assuming that a Gaussian controller is parameterized by a function space $\mathcal{H}$,

$$p(\mathbf{a}|\mathbf{s}; h) \sim \mathcal{N}(h(\mathbf{s}); \Sigma) \tag{1}$$

where $\mathbf{a} \in \mathcal{A}$ is an action, $\mathbf{s} \in \mathcal{S}$ is a state, and assuming $\Sigma = \sigma^2\mathbf{I}$. For parametric policy approaches, $h$ may be a neural network or a linear function of predefined features as

$$h(\mathbf{s}) = \sum_{i=1}^{n} w_i \phi_i(\mathbf{s}) \tag{2}$$

The authors in (Heidrich-Meisner and Igel, 2009a) propose to use the CMA-ES algorithm to optimize a cumulative reward objective function,

$$J(\theta) = \mathbb{E}_{\pi(\theta)} \left( \sum_{t=0}^{T} \gamma^t r_t \right) \tag{3}$$

For each candidate $\theta_i$, its evaluation is $J(\theta_i)$ which is computed using Monte-Carlo simulations. Specifically, a set of $N$ trajectories are sampled by executing $\pi(\theta_i)$, hence $J(\theta_i) \approx \frac{1}{N}\sum_{j=1}^{N} R(\tau_j)$ where $R(\tau_j)$ is a return of trajectory $j$.

#### 2.2.2. Policy gradient in RKHS

These work (Bagnell and Schneider, 2003) and (Lever and Stafford, 2015) suggests to represent policies in reproducing kernel Hilbert spaces, where the function $h(\cdot)$ in Eq. 1 is in a vector-valued function in RKHS $\mathcal{H}_K$,

$$h(\cdot) = \sum_{i} K(\mathbf{s}_i, \cdot)\alpha_i, \qquad \text{where } \alpha_i \in \mathcal{A} \tag{4}$$

where $K$ is a vector-valued kernel, $K : \mathcal{S} \times \mathcal{S} \mapsto \mathcal{L}(\mathcal{A})$ (Micchelli and Pontil, 2005), in which $\mathcal{L}(\mathcal{A})$ is the space of linear operators on $\mathcal{A}$. Using this functional policy parametrization, we can analytically derive functional gradients by using the Fréchet derivative, and compute its approximate value given a trajectory set $\{\tau_i\}_{i=1}^{N}$, each of length $H$,

$$\begin{aligned} \nabla_h J(\pi_h) &\approx \frac{1}{N} \sum_{i=1}^{N} \nabla_h \log P(\tau_i; h) R(\tau_i) \\ &= \frac{1}{N} \sum_{i=1}^{N} \sum_{t=0}^{H} Q(\mathbf{s}_t, \mathbf{a}_t) K(\mathbf{s}_t, \cdot) \Sigma^{-1} (\mathbf{a}_t - h(\mathbf{s}_t)) \end{aligned} \tag{5}$$

where $Q(\mathbf{s}_t, \mathbf{a}_t)$ is a sample of the Q-value function of $(\mathbf{s}_t, \mathbf{a}_t)$ (this equality results from the Policy Gradient theorem (Sutton et al., 1999)). Thus, $\nabla_h J(\pi_h)$ is also a function in $\mathcal{H}_K$, and the functional gradient update at iteration $k$ is

$$h_{k+1} \leftarrow h_k + \alpha \nabla_h J(\pi_{h_k}) \tag{6}$$

where $\alpha$ is a step-size. As the representation of each policy $h_{k+1}$ becomes more complex after each iteration (Lever and Stafford, 2015), a sparsification technique is often used to achieve a sparse and adaptive policy. On the other hand, a compatible kernel for Q-functions $Q^\pi(\mathbf{s}, \mathbf{a})$ can simply be derived based on kernel $K$, that is $K_h$ as

$$K_h\big((\mathbf{s}, \mathbf{a}), (\mathbf{s}', \mathbf{a}')\big) = \Big(K(\mathbf{s}, \mathbf{s}')\Sigma^{-1}(\mathbf{a} - h(\mathbf{s}))\Big)^\top \Sigma^{-1}(\mathbf{a}' - h(\mathbf{s}')) \tag{7}$$

Kernel regression methods can be used to approximate $Q$ easily via kernel matching pursuit (Vincent and Bengio, 2002), k-LSTDQ (Xu et al., 2007), etc. This framework is called the compatible RKHS Actor-Critic framework, RKHS-AC, by (Lever and Stafford, 2015). Though RKHS-AC has very excellent policy modeling, it would only result in locally optimal policies. A global policy search technique like CMA-ES which may be considered as a powerful off-the-shelf tool, its current form has many drawbacks. For many applications of CMA-ES, a parametric objective function is often designed, which requires a set of predefined feature function. Hence, the choice of features plays a very important role. Recently, the authors in (Vien et al., 2016) propose a natural actor-critic in RKHS algorithm by computing the Fisher information operator, which is analogue to the Fisher information matrix for parametric policies.

## 3. CMA-ES in Reproducing Kernel Hilbert Space

### 3.1. Problem statement

We consider a functional optimization problem (Ulbrich, 2009) that finds the maximum of an unknown functional $f : \mathcal{H} \mapsto \Re$, where $\mathcal{H} = \{h : \mathcal{X} \mapsto \Re\}$ is a separable Hilbert space of input real-valued functions with a domain $\mathcal{X}$. Assuming that $\mathcal{H}$ is specifically a square-integrable function space $\mathcal{L}^2(\mathcal{X}, \mu)$ w.r.t a probability measure $\mu$. For each queried function $h \in \mathcal{H}$, an evaluation $y = f(h)$ is returned.

### 3.2. CMA-ES in RKHS

We propose a new general-purposed CMA-ES-RKHS framework that solves the above problem. We explicitly assume the function space $\mathcal{H}$ is a reproducing kernel Hilbert space (RKHS) associated with a kernel $K$. Each $h \in \mathcal{H}$ is defined as a mapping from an arbitrary space $\mathcal{X}$ to $\mathcal{Y}$, $h : \mathcal{X} \mapsto \mathcal{Y}$. The function space $\mathcal{H}$ may be a vector-valued RKHS (Micchelli and Pontil, 2005), denoted as $\mathcal{H}_K$, with the kernel $K : \mathcal{X} \times \mathcal{X} \mapsto \mathcal{L}(\mathcal{Y})$, where $\mathcal{L}(\mathcal{Y})$ is the space of linear operators on $\mathcal{Y}$. For example, when $\mathcal{X} = \Re^n$ the simplest choice of $K$ might be $K(x, x') = \kappa(x, x')\mathrm{I}_n$, where $\mathrm{I}_n$ is an $n \times n$ identity matrix, and $\kappa$ is a scalar-valued kernel (Schölkopf and Smola, 2002). Each function $h \in \mathcal{H}$ is then represented as a linear span of finite elements $\{x_i, y_i\}$ as

$$h(\cdot) = \sum_i K(x_i, \cdot) y_i \tag{8}$$

Now we define a search distribution over $\mathcal{H}$. A direct extension of parametric CMA-ES is to use a Gaussian process over the solution function $h$, $h \sim \mathcal{GP}(m, \sigma^2 C)$, where $m$ is a mean function, $C$ is a covariance operator, and $\sigma$ is a scalar global step-size. We discuss now how to update the functionals $\{m, C\}$ and the parameter $\sigma$ in our CMA-ES-RKHS framework, which is also summarvvized in Algorithm 2.

#### 3.2.1. Mean function update in RKHS

Assuming that at iteration $k$, we can sample a set of $\lambda$ functions $\tilde{g}_i \sim \mathbb{GP}(0, C)$ (Step 4), many sampling techniques are basically described in (Rasmussen and Williams, 2006). Our following derivation is not restricted to which kernel, vector-valued or scalar-valued, is used.

**Algorithm 2** The CMA-ES-RKHS algorithm

1: Initialize $m \in \mathcal{H}_K$, $\sigma \in \Re^+$, $\lambda, \mu$
2: Initialize $C = \delta(\cdot,\cdot), p_c \in \mathcal{H}_K, p_\sigma \in \mathcal{H}_K$
3: **while** (not terminate) **do**
4: *Sampling*: $\tilde{g}_i \sim \mathcal{GP}(0, C), i = 1, \ldots, \lambda$
5: Via kernel regression: for each $\tilde{g}_i$, fit a function $g_i \in \mathcal{H}_K$
6: Set samples: $h_i = m + \sigma^{(t)} g_i$
7: *Evaluating*: $f_i = f(h_i), i = 1, \ldots, \lambda$
8: // *mean update*
9: $m \leftarrow m + \sigma\bar{g}$, where $\bar{g} = \sum_1^\mu w_i g_{i:\lambda}$
10: // *step-size control update*
11: $p_\sigma \leftarrow (1 - c_\sigma)p_\sigma + \sqrt{c_\sigma(2 - c_\sigma)\mu_w} C^{-\frac{1}{2}} \bar{g}$
12: $\sigma \leftarrow \sigma \exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{\|p_\sigma\|}{\mathbb{E}\|\mathcal{GP}(0,\delta(\cdot,\cdot))\|} - 1\right)\right)$
13: // *covariance matrix update*
14: $p_c \leftarrow (1 - c_c)p_c + \sqrt{c_c(2 - c_c)\mu_w}\bar{g}$
15: $C \leftarrow (1 - c_1 - c_\mu)C + c_1 p_c \otimes p_c + c_\mu \sum_1^\mu w_i g_{i:\lambda} \otimes g_{i:\lambda}$
16: Sparsify $m, C$ and derive $C^{-\frac{1}{2}}$
17: **end while**

A sample from a Gaussian process is not in $\mathcal{H}_K$ with probability of 1, as discussed in detail by (Adler, 1981). For any sampling techniques of a Gaussian process, we receive $\tilde{g}_i$ in a form of data tuples $(x^{(i)}, y^{(i)})$ from which we can use kernel ridge regression with the covariance operator $C(\cdot,\cdot)$ (Step 5). Hence, in our framework each function $\tilde{g}_i$ is approximated by a function $g_i \in \mathcal{H}_K$. As a result, a new function candidate sampled from the function distribution is $h_i = m + \sigma g_i$. The new mean function is updated as (Step 9),

$$m = m + \sigma \sum_{i=1}^{\mu} w_i g_{i:\lambda} \in \mathcal{H}_K \tag{9}$$

where the normalized weights $w_i$ satisfy

$$\sum_{i=1}^{\mu} w_i = 1, \quad w_1 \geq w_2 \geq \cdots \geq w_\mu > 0$$

As a result, after the update the new functional mean is an element in $\mathcal{H}_K$. We denote $\bar{g}$ as

$$\bar{g} = \sum_{i=1}^{\mu} w_i g_{i:\lambda}$$

There are a number of settings for $w$, which might inherit from CMA-ES, such as: $w_i = 1/\mu$, $w_i \propto \mu - i + 1$; or a better choice $w_i = \log(\mu + \frac{1}{2}) - \log(i)$. In our experiment, we implement the last choice.

### 3.2.2. Covariance operator update

The covariance operator update is based on the best selected candidate functions, based on their evaluations $f(h_i)$. Hence an empirical estimate of the covariance operator $C$ on $\mathcal{H}_K$,

called *rank-$\mu$ update*, is

$$C = (1 - c_\mu)C + c_\mu \sum_{i=1}^{\mu} w_i g_{i:\lambda} \otimes g_{i:\lambda}$$

Similar to parametric CMA-ES, we also consider the change of the mean function over time by estimating a functional evolution path $p_c$ as (Step 14),

$$p_c = (1 - c_c)p_c + \sqrt{c_c(2 - c_c)\mu_w}\bar{g} \in \mathcal{H}_K \tag{10}$$

where $\mu_w$ is a variance-effectiveness constant. This is low-pass filtered of chosen steps $\bar{g}$, hence $p_c$ is also an element in RKHS $\mathcal{H}_K$. As a result, a complete update of the covariance operator that combines both rank-1 and rank-$\mu$ is computed as (Step 15),

$$C = (1 - c_\mu - c_1)C + c_1 p_c p_c^\top + c_\mu \sum_{i=1}^{\mu} w_i g_{i:\lambda} \otimes g_{i:\lambda} \tag{11}$$

where $c_c$ is the backward time horizon for the functional evolution path $p_c$, and $c_1, c_\mu$ are learning rates of rank-1 and rank-$\mu$ respectively. This reduces to a rank-1 update if $c_1 = 1, c_\mu = 0$. Similarly, the update becomes a rank-$\mu$ update when $c_1 = 0, c_\mu = 1$.

#### 3.2.3. Step-size update

The global step-size $\sigma$ is adapted through the computation of a functional conjugate evolution path as (Step 11),

$$p_\sigma = (1 - c_\sigma)p_\sigma + \sqrt{c_\sigma(2 - c_\sigma)\mu_{\mathrm{w}}}\mathrm{C}^{-\frac{1}{2}}\bar{g} \tag{12}$$

where $c_\sigma$ is a backward time horizon for the conjugate evolution path $p_\sigma$. According to the bounded inverse theorem in functional analysis (Conway, 2013), $C$ as computed in Eq. 11 is a linear operator in the RKHS $\mathcal{H}_K$, hence it has a bounded inverse $C^{-1}$. Therefore, $p_\sigma$ is updated in a way that renders it an element in $\mathcal{H}_K$. The volume and the correlation of the selected steps are compared to the expected value of the standard Gaussian process with a Dirac kernel. The fact that the former is larger than the latter makes $\sigma$ increased, otherwise decreased. The update formula of $\sigma$ (Step 12) is

$$\sigma = \sigma \exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{\|p_\sigma\|}{\mathbb{E}\|\mathcal{GP}(0, \delta(\cdot,\cdot))\|} - 1\right)\right) \tag{13}$$

The term $\mathbb{E}\|\mathcal{GP}(0, \delta_x)\|$ can be computed in advance using Monte-Carlo simulations

$$\mathbb{E}\|\mathcal{GP}(0, \delta(\cdot,\cdot))\|_{\mathcal{H}_K} \approx \frac{1}{N}\sum_{i=1}^{N}\langle g_i(\cdot), g_i(\cdot)\rangle_{\mathcal{H}_K}$$

where $g_i(\cdot)$ is a function in $\mathcal{H}_K$ approximated (via kernel ridge regression) from a sample $\tilde{g}_i$ drawn from $\mathcal{GP}(0, \delta(\cdot,\cdot))$.

#### 3.2.4. Sparsification and Adaptive Representation

We now discuss implementation concerns of the CMA-ES-RKHS algorithm. The first and most critical one is the representation issue of mean functions $m$ and covariance operators $C$. Then, it follows with discussions of parameter setting in CMA-ES-RKHS. Then we discuss how to deal with the update rule in Eq. 12 that involves to find the inverse operator $C^{-\frac{1}{2}}$.

**Sparsification** (Step 16): The updates of the mean function in Eq. 9 and covariance operator in Eq. 11 make their representation complexity increase linearly on the number of iterations. Though we receive an adaptive, flexible and complex policy, this would result in an expensive evaluation cost, e.g. computing $m(x)$ or $C(h, h')$ when needed, where $h, h' \in \mathcal{H}_K$. Sparsification is a technique that is able to keep these kernel-based representation sparse and approximately accurate. Assuming that after each iteration, $m$ and $C$ are re-written in forms of

$$m = \sum_i^{N_1} \beta_i K(x_i, \cdot), \quad C = \sum_i^{N_2} \lambda_i h_i \otimes h_i \tag{14}$$

where $x_i \in \mathcal{X}, h_i \in \mathcal{H}_K$, and $\beta_i, \lambda_i \in \mathcal{Y}$. A sparsification algorithm would sparsify $m$ and $C$ to become

$$m = \sum_i^{n_1} \tilde{\beta}_i K(\tilde{x}_i, \cdot), \quad C = \sum_i^{n_2} \tilde{\lambda}_i \tilde{h}_i \otimes \tilde{h}_i$$

where $\tilde{x}_i \in \mathcal{X}, \tilde{h}_i \in \mathcal{H}_K$ and $n_1 \ll N_1, n_2 \ll N_2$. In our CMA-ES-RKHS framework, we resort to two different sparsification techniques separately for $m$ and $C$. We propose to use the kernel matching pursuit algorithm (Vincent and Bengio, 2002) to sparsify $m$. Its idea is to add kernel features $K(x_i, \cdot)$ sequentially and greedily that maximally reduce the current approximation error. A tolerance constant is used to check the error reduction level before adding a new kernel feature. The use of tolerance also helps achieving more compact representation.

In general, we can use the kernel matching pursuit algorithm (Vincent and Bengio, 2002) to sparsify $C$. However, we aim to look for a method that will both sparsify $C$ and together compute the inverse square root operator $C^{-\frac{1}{2}}$. Therefore, we propose to use the *kernel PCA* method (kPCA) from (Schölkopf et al., 1997) for achieving efficiently and fast both a sparse and compact covariance operator and its inverse square root operator. Specifically, we rewrite $C$ in Eq. 14 as

$$C = H \Lambda H^\top = \Phi \Phi^\top$$

where $H$ is a matrix whose $i$th column is $h_i$, a $N_2 \times N_2$ diagonal matrix $\Lambda = \text{diag}(\lambda_i)$, and $\Phi = H\Lambda^{\frac{1}{2}}$. Via kPCA, $C$ can be decomposed through a decomposition of the $N_2 \times N_2$ Gram matrix $G = \Phi^\top \Phi$, in which $G(i, j) = \sqrt{\lambda_i \lambda_j} \langle h_i, h_j \rangle_{\mathcal{H}_K}$. If a singular value decomposition (SVD) of $G$ is $G = UDU^\top$, the decomposition of $C$ via kPCA is $C = VDV^\top$, where $V = \Phi U D^{-\frac{1}{2}}$ are orthonormal eigenfunctions of $C$, hence $V = H\Lambda^{\frac{1}{2}} U D^{-\frac{1}{2}}$. One could easily show that each eigenfunction of $C$ is a linear span of $\{h_i\}_{i=1}^{N_2}$,

$$v_i = \sum_i w_{ij} h_j(\cdot)$$

where $w_{ij}$ is an element of a matrix $W = \Lambda^{\frac{1}{2}} U D^{-\frac{1}{2}}$. Hence $v_i$ is also an element in $\mathcal{H}_K$.

From this kPCA of $C$, we are now able to sparsify $C$ by choosing only a small set of eigenfunctions of principal eigenvalues. Moreover, from the decomposition of $C$, the inverse square root operator $C^{-\frac{1}{2}}$ is derived as

$$C^{-\frac{1}{2}} = V D^{-\frac{1}{2}} V^\top \tag{15}$$

which is a linear operator in RKHS $\mathcal{H}_K$.

**Parameter Setting**: The mean function is represented by $n_1$ adaptive kernel features, hence it has $n_1$ pivotal parameters. In CMA-ES, parameter setting is based on the number of free parameters, the dimension of the search space. Though it is not precise, we use the same setting of CMA-ES for CMA-ES-RKHS, i.e. the parameters: $c_1, c_c, c_\mu, c_\sigma, d_\sigma$ based on $n_1$. We call $n_1$ the *effective dimensionality* on our CMA-ES-RKHS.

### 3.3. EDA and CEM in RKHS

For a short notice, the derivations of CMA-ES-RKHS can easily be transferred to derive EDA-RKHS and CEM-RKHS algorithms. Specifically, the updates at iteration $k$ for EDA-RKHS and CEM-RKHS change step 9 and 15 in Algorithm 2 as follows

$$\begin{aligned} m^{(k)} &= \frac{1}{\mu} \sum_{i=1}^{\mu} h_i \in \mathcal{H}_K \\ C^{(k)}_{\text{EDA-RKHS}} &= \frac{1}{\mu} \sum_{i=1}^{\mu} (h_i - m^{(k)}) \otimes (h_i - m^{(k)}) \\ C^{(k)}_{\text{CEM-RKHS}} &= \frac{\mu}{\mu - 1} C^{(k)}_{\text{EDA-RKHS}} \end{aligned}$$

though the updates of $C^{(k)}_{\text{EDA-RKHS}}$ and $C^{(k)}_{\text{CEM-RKHS}}$ are simpler than $C$ of CMA-ES-RKHS, they result in similar covariance operators on $\mathcal{H}_K$. Hence the implementation technique of EDA-RKHS and CEM-RKHS can be similar to that of CMA-ES-RKHS which has been discussed above. Therefore, we will not put them in comparisons.

## 4. Direct Policy Search via CMA-ES-RKHS

In RL, there is recent effort to model policies as functions in RKHS (Bagnell and Schneider, 2003; Lever and Stafford, 2015; Vien et al., 2016). Such RKHS policy gradient approaches suffer from a problem of step-size. Though an extended work, RKHS EM-based policy search (RKHS-PoWER) by (Vien et al., 2016), would overcome this issue, it can only converge to local optima. We propose a new black-box direct policy search in RKHS that is based on CMA-ES-RKHS (the extension to EDA-RKHS and CEM-RKHS is similar). The policy is a function in RKHS with a kernel $K$ as first introduced in Eq. 4. Different from the parametric CMA-ES direct policy search (Heidrich-Meisner and Igel, 2009a), our non-parametric modeling enables optimization in a very rich policy space and allows to learn more complex policies that also enjoy adaptive and compact representation and do not depend on fixed features.

**Adaptive CMA-ES Direct Policy Search**: There is a naive way that modifies the parametric CMA-ES direct policy search (Heidrich-Meisner and Igel, 2009a) to become adaptive, which will be used as base-line to compare to CMA-ES-RKHS policy search. Assuming that we use a controller which is a linear function of RBF features $\phi_i(\mathbf{s}_i)$ in which $\mathbf{s}_i$ is a centre,

$$h(s) = \sum_{i=1}^{n} w_i \phi_i(\mathbf{s}_i) \tag{16}$$

Applying CMA-ES direct policy search method (Heidrich-Meisner and Igel, 2009a), the parameter space would be $\mathbf{w} = \{w_i\} \in \Re^n$. We make a slight change to assume that the parameter space is $\{w_i, \mathbf{s}_i\}$, called *adaptive CMA-ES direct policy search* (CMA-ES-A). A clear problem of CMA-ES-A is in determining the scaling of these parameters. However it is in principle overcome by the RKHS norm on functions as in our CMA-ES-RKHS algorithm.

## 5. Experiments

We first evaluate the advantages and general applications of CMA-ES-RKHS on two simple functional optimization problems: 1-D and 2-D function spaces. We compare the behavior of CMA-ES-RKHS with other three base-line methods: the standard CMA-ES, the adaptive CMA-ES version (CMA-ES-A), and the functional gradient techniques. The next experiments are two RL tasks, inverted pendulum and double-pendulum. We compare our direct policy search via CMA-ES-RKHS to the standard CMA-ES policy search, the adaptive CMA-ES policy search, a parametric actor-critic, and the actor-critic in RKHS (RKHS-AC) methods. In all experiments, we use the RBF kernel where the bandwidths are set using *median-trick*. These experiments aim to evaluate the proposed CMA-ES-RKHS for: i) the quality of the returned compact solution function, ii) the flexibility and power of the proposed method in capturing a complex solution function which can not be found easily by existing methods, iii) the applicability in practice, i.e. for direct policy search.

### 5.1. Synthetic Domains

We design two unknown 1-D and 2-D functions $f^*$. Each function is a mixture of two (multivariate) Gaussians, respectively. All optimizers are tasked to find a function $h : \mathcal{X} \mapsto \Re$, where $h \in \mathcal{H}_K$ that minimizes the objective function as a square distance

$$J(h) = \int_{x_0}^{x_T} \big(f^*(x) - h(x)\big)^2 dx$$

where $x \in \Re^k$, $k = 1, 2$ correspondingly to the 1-D or 2-D domain. This task is a simplified version of many similar problems in machine learning and robotics, e.g. regularized risk functional (Schölkopf and Smola, 2002), trajectory optimization (Toussaint, 2014), trajectory optimization in RKHS (Marinho et al., 2016), loss minimization inverse optimal control (Doerr et al., 2015), etc.. However, these work must rely on discretization and parametric modeling.

**Functional gradient**: Using functional gradient requires to know $J$ and have access to the ground-truth function $f^*$ (CMA-ES-RKHS only accesses to evaluations $J(h)$) from

which we are able use discretization to approximate $J$ as

$$J(h) \approx \frac{1}{T}\sum_{k=0}^{T}\big(f^*(x_k) - h(x_k)\big)^2$$

The functional gradient can be computed as: $\nabla_h J(h) = \sum_{k=0}^{T} 2\big(h(x_k) - f^*(x_k)\big)K(t_k, \cdot)$. Thus, a functional gradient update is $h \leftarrow h + \alpha\nabla_h J(h)$. A sparsification technique (Vincent and Bengio, 2002) can be used to achieve a compact representation of $h$ which renders the functional gradient approach an adaptive method too. That means the representation of $h$ will be adaptively adapted to best approximate $f^*$. Hence, discretization is required to be fine enough (T is large enough, we used $T \gg N$) to guarantee accurate approximation.

**CMA-ES**: We assume that a parametric representation of $h$ as a linear expansion of $N$ features:$h(x) = \sum_{k=1}^{N} w_k\phi_k(x) = \mathbf{w}^\top\phi(x)$. We use RBF features $\phi_k(x) = \exp(-\|x - x_t\|^2/\sigma^2)$ in which $N$ centers $x_t$ are regular intervals in the domain of $x$. Hence we apply CMA-ES to optimize $J$ in a parameter space $\mathbf{w} \in \Re^N$. CMA-ES-A would optimize over a search space of $\{\mathbf{w}, \{x_t\}_{t=1}^N\}$.

**Results**: For all optimizers, we use the same number $N$ of features in CMA-ES and CMA-ES-A, and centres after sparsification in CMA-ES-RKHS and functional gradient methods. We use $N = 10$ for 1-D task, and $N = 100$ for 2-D task. As mentioned in parameter setting section, we use a standard way of CMA-ES to initialize other parameters, $N$ is the effective dimensionality in CMA-ES-RKHS. The results are reported w.r.t the number of evaluations, i.e queries to the objective function.

We report the squared error $J$ and the solution function for the 1-D task in Fig. 1 and the 2-D task in Fig. 2 and Fig. 3 (on the left). We create two versions for CMA-ES-A, one with good initialization (initial values of $x_t$ are centres for CMA-ES) and one with random initialization, called CMA-ES-A-R. In Fig. 1, the performance of CMA-ES-A-R is very bad in terms of error. As demonstrated on the right picture, it can detect only one mode of the optimal function. Hence we give up reporting results from CMA-ES-A-R in other domains. One remarkable note is that CMA-ES initialization does not consists of two correct modes in its set of centres, hence it gives poor approximation error. With adaptive ability, CMA-ES-A and CMA-ES-RKHS are able to estimate the true modes correctly.

In the larger domain (2-D), CMA-ES-A performs much worse than our method. This is explained by the way our method approaches from a principled way, i.e kernel methods, for the scaling of parameters. The functional gradient method performs very well which re-confirms that it can be very competitive when gradient information is known (in this case the form of $J(h)$ is known). Fig. 2 shows very interesting results where other methods like CMA-ES and CMA-ES-A are still struggling around the optimal regions.

### 5.2. Inverted Pendulum

We use the same setting of the inverted pendulum domain as in (Lever and Stafford, 2015). This problem has an 1-dim action space $[-3, 3]$, a state space $s = (\theta, \omega)$, where $\theta \in [-\pi, \pi]$ is angular position and $\omega \in [-4\pi, 4\pi]$ is angular velocity. The system always starts at $s_0 = (-\pi, 0)$ (downward position). The reward function is $r(s, a) = \exp(-0.5\theta^2)$ that requires to bring the pole to the upright position and keep it balanced there. The dynamics of the system is $\theta' = \theta + 0.05\omega + \epsilon$; $\omega' = \omega + 0.05a + 2\epsilon$, $\epsilon$ is a small Gaussian noise

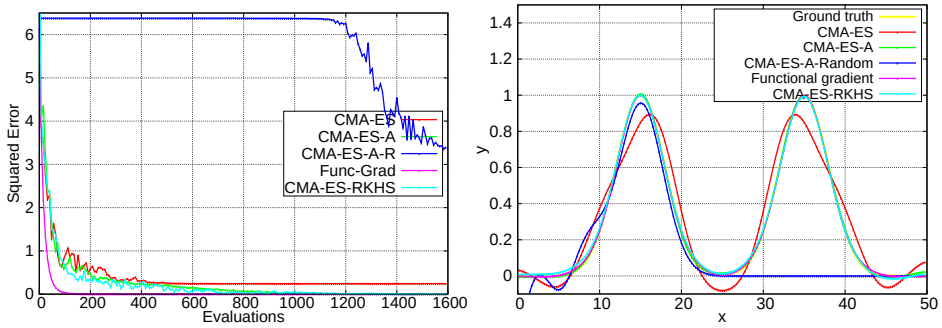


Figure 1: 1D synthetic domain: (left) squared error, (right) solution functions

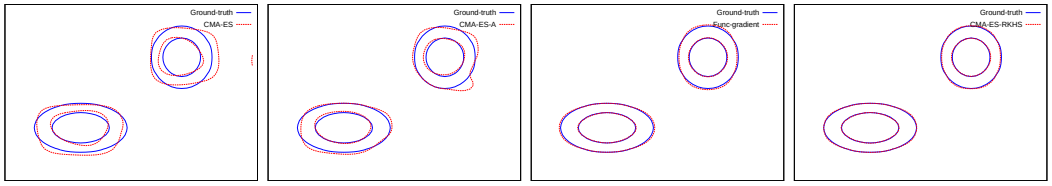


Figure 2: 2D synthetic domain: contours of levels equivalent to the first and second deviations

$\mathcal{N}(0, 0.02^2)$. We use $N = 50$ centres or features for all algorithms. We set $\gamma = 0.99$ and a horizon $H = 400$.

The results of mean performance and its 95% confidence are computed over 15 runs and reported in Fig. 3 (on the center). In this task, CMA-ES performs better than CMA-ES-A and CMA-ES-RKHS. CMA-ES-A has a much bigger search space comparing to that of CMA-ES, $3N$ vs. $N$ parameters. We conjecture that CMA-ES-RKHS performs worse because we use $N$ as the effective dimensionality to set its parameters. Theoretically, CMA-ES-RKHS optimizes over a potentially infinite dimensional space. We tried to increase its effective dimensionality to $2N$, called Eff. CMA-ES-RKHS. This modification improves the performance significantly. CMA-ES-A-R performs slowly but keeps improving constantly. Local direct policy search algorithms, AC and RKHS-AC, do not perform comparably to the others.

### 5.3. Double Pendulum

This problem consists of two links and two under-actuated joints. The system state is 4-dimensional of joint position and velocities $\mathbf{s} = \{\theta_1, \dot{\theta}_1, \theta_2, \dot{\theta}_2\}$. Actions are motor torques $\mathbf{a} = [u_1, u_2]$, which are limited in $[-5N, 5N]$. The dynamics is simulated using second-order Runge-Kutta. We use a low sampling frequency of $50Hz$ at which torques could be applied. The start state is $\{0, -\pi, 0, 0\}$. The reward function is $r(\mathbf{s}) = \exp(-\|\mathbf{s} - \mathbf{s}^*\|_W)$, where $\mathbf{s}^* = \{0, 0, 0, 0\}$ and $W = \text{diag}(0.25, 0.0025, 0.25, 0.0025)$. Each episode is simulated in $6s$, which is equivalent to a horizon of 300 steps. We use $N = 256$ features or centres, and $\gamma = 0.99$. The optimal policy returns 88. We only compare between global policy search methods via CMA-ES, CMA-ES-A, and CMA-ES-RKHS. In this complex task, CMA-ES-RKHS has clearly outperformed other methods as seen in Fig. 3 (right).

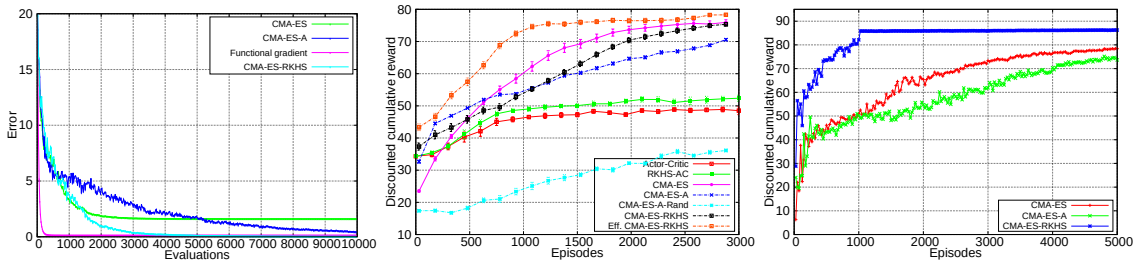


Figure 3: Results for (left) 2D synthetic domain, (center) the inverted pendulum domain; (right) Double-link

## 6. Conclusion

This paper proposes a CMA-ES-RKHS framework that enables functional optimization where the search is handled over a function space. The fact that the function space is modeled in reproducing kernel Hilbert space results in analytic update rules for CMA-ES-RKHS. On the other hand, the solution function attains compactness and flexibility characteristics. We apply CMA-ES-RKHS for direct policy search in which the policy is modeled in RKHS. Our experiments show that both CMA-ES-RKHS and direct policy search via CMA-ES-RKHS are able to represent a complex solution function compactly and adaptively. The result shows many interesting aspects and results of CMA-ES-RKHS: i) explicitly handling functional optimization in principle; ii) overcoming the issue of hand-designed feature functions in many practical applications of CMA-ES. More practical applications of CMA-ES-RKHS would be a very interesting future research direction.

## Acknowledgement

This research is funded by Vietnam National Foundation for Science and Technology Development (NAFOSTED) under grant number 102.05-2016.18; and by the Basic Science Research Program through National Research Foundation of Korea (NRF) of the Ministry of Education, Science, and Technology under grant number 2014R1A1A2057735.

## References

Robert J. Adler. *The Geometry of Random Fields*. John Wiley & Sons, 1981.

J. Andrew (Drew) Bagnell and Jeff Schneider. Policy search in reproducing kernel hilbert space. Technical Report CMU-RI-TR-03-45, Robotics Institute, Pittsburgh, PA, November 2003.

John B Conway. *A course in functional analysis*, volume 96. Springer Science & Business Media, 2013.

Andreas Doerr, Nathan D. Ratliff, Jeannette Bohg, Marc Toussaint, and Stefan Schaal. Direct loss minimization inverse optimal control. In *Robotics: Science and Systems XI*, 2015.

Sehoon Ha and C. Karen Liu. Iterative training of dynamic skills inspired by human coaching techniques. *ACM Trans. Graph.*, 34(1):1:1–1:11, 2014.

Sehoon Ha and C. Karen Liu. Evolutionary optimization for parameterized whole-body dynamic motor skills. In *ICRA*, pages 1390–1397, 2016.

Nikolaus Hansen. The CMA evolution strategy: A tutorial. *CoRR*, abs/1604.00772, 2016.

Nikolaus Hansen and Anne Auger. Principled design of continuous stochastic search: From theory to practice. In *Theory and principled methods for the design of metaheuristics*, pages 145–180. Springer, 2014.

Nikolaus Hansen, Sibylle D. Müller, and Petros Koumoutsakos. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). *Evolutionary Computation*, 11(1):1–18, 2003.

Verena Heidrich-Meisner and Christian Igel. Similarities and differences between policy gradient methods and evolution strategies. In *ESANN*, pages 149–154. Citeseer, 2008.

Verena Heidrich-Meisner and Christian Igel. Neuroevolution strategies for episodic reinforcement learning. *J. Algorithms*, 64(4):152–168, 2009a.

Verena Heidrich-Meisner and Christian Igel. Hoeffding and bernstein races for selecting policies in evolutionary direct policy search. In *ICML*, pages 401–408, 2009b.

Guy Lever and Ronnie Stafford. Modelling policies in mdps in reproducing kernel hilbert space. In *AISTATS*, 2015.

Shie Mannor, Reuven Y. Rubinstein, and Yohai Gat. The cross entropy method for fast policy search. In *Machine Learning, Proceedings of the Twentieth International Conference (ICML 2003), August 21-24, 2003, Washington, DC, USA*, pages 512–519, 2003.

Zita Marinho, Byron Boots, Anca Dragan, Arunkumar Byravan, Geoffrey J. Gordon, and Siddhartha Srinivasa. Functional gradient motion planning in reproducing kernel hilbert spaces. In *RSS*, 2016.

Charles A. Micchelli and Massimiliano Pontil. On learning vector-valued functions. *Neural Computation*, 17(1):177–204, 2005.

Jan Peters and Stefan Schaal. Natural actor-critic. *Neurocomputing*, 71(7):1180–1190, 2008.

Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

Reuven Rubinstein. The cross-entropy method for combinatorial and continuous optimization. *Methodology and computing in applied probability*, 1(2):127–190, 1999.

Reuven Y Rubinstein and Dirk P Kroese. *The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation and machine learning*. Springer Science & Business Media, 2013.

Elmar A. Rückert, Gerhard Neumann, Marc Toussaint, and Wolfgang Maass. Learned graphical models for probabilistic planning provide a new class of movement primitives. *Front. Comput. Neurosci.*, 2013, 2013.

Bernhard Schölkopf and Alexander J. Smola. *Learning with Kernels Support Vector Machines, Regularization, Optimization, and Beyond*. Adaptive Computation and Machine Learning series, MIT Press, 2002.

Bernhard Schölkopf, Alexander J. Smola, and Klaus-Robert Müller. Kernel principal component analysis. In *ICANN*, pages 583–588, 1997.

Freek Stulp and Olivier Sigaud. Path integral policy improvement with covariance matrix adaptation. In *ICML*, 2012.

Richard S. Sutton, David A. McAllester, Satinder P. Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *NIPS*, pages 1057–1063, 1999.

Jie Tan, Yuting Gu, Greg Turk, and C Karen Liu. Articulated swimming creatures. *ACM Transactions on Graphics (TOG)*, 30(4):58, 2011.

Marc Toussaint. Newton methods for k-order Markov constrained motion problems. *CoRR*, abs/1407.0414, 2014.

Le Pham Tuyen, Ngo Anh Vien, and P.Marlith Jaramillo TaeChoong Chung. Importance sampling policy gradient algorithms in reproducing kernel hilbert space. *Artificial Intelligence Review*, 2017. To Appear.

Michael Ulbrich. Optimization methods in Banach spaces. In *Optimization with PDE Constraints*, pages 97–156. Springer, 2009.

H. van Hoof, J. Peters, and G. Neumann. Learning of non-parametric control policies with high-dimensional state features. In *(AISTATS*, 2015.

Ngo Anh Vien, Peter Englert, and Marc Toussaint. Policy search in reproducing kernel hilbert space. In *IJCAI*, pages 2089–2096, 2016.

Pascal Vincent and Yoshua Bengio. Kernel matching pursuit. *Machine Learning*, 48(1-3): 165–187, 2002.

Jack M. Wang, Samuel R. Hamner, Scott L. Delp, and Vladlen Koltun. Optimizing locomotion controllers using biologically-based actuators and objectives. *ACM Trans. Graph.*, 31(4):25:1–25:11, 2012.

Daan Wierstra, Tom Schaul, Tobias Glasmachers, Yi Sun, Jan Peters, and Jürgen Schmidhuber. Natural evolution strategies. *Journal of Machine Learning Research*, 15(1):949–980, 2014.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

Xin Xu, Dewen Hu, and Xicheng Lu. Kernel-based least squares policy iteration for reinforcement learning. *IEEE Trans. Neural Networks*, 18(4):973–992, 2007.