A Functional Optimization Method for Continuous Domains

Viet-Hung Dang[1], Ngo Anh Vien[2(✉)], Pham Le-Tuyen[3], and Taechoong Chung[3]

[1] Research and Development Center, Duy Tan University, Da Nang, Vietnam
[2] Machine Learning and Robotics Lab, University of Stutgart, Stuttgart, Germany
vien.ngo@gmail.com
[3] Department of Computer Engineering, Kyung Hee University, Seoul, Korea

Abstract. Smart city solutions are often formulated as adaptive optimization problems in which a cost objective function w.r.t certain constraints is optimized using off-the-shelf optimization libraries. Covariance Matrix Adaptation Evolution Strategy (CMA-ES) is an efficient derivative-free optimization algorithm where a black-box objective function is defined on a parameter space. This modeling makes its performance strongly depends on the quality of chosen features. This paper considers modeling the input space for optimization problems in reproducing kernel Hilbert spaces (RKHS). This modeling amounts to functional optimization whose domain is a function space that enables us to optimize in a very rich function class. Our CMA-ES-RKHS framework performs black-box functional optimization in the RKHS. Adaptive representation of the function and covariance operator is achieved with sparsification techniques. We evaluate CMA-ES-RKHS on simple functional optimization problems which are motivated from many problems of smart cities.

Keywords: Functional optimization · Smart city · Cross-entropy
Covariance matrix adaptation evolution strategy

1 Introduction

The smart city initiative uses information and communication technology (ICT) and Internet of things (IoT) solutions to manage a city's assets, e.g. transportation systems, grid networks, schools, hospitals, etc. The goal is to build a smart city that improves the quality of life and the efficiency of services. These goals are often realized as a cost objective function of energy, price, consumption, user's comfort, and so on [3]. Optimizing such an objective can be handled via various optimization libraries depending on different situations. The covariance matrix adaptation evolutionary strategy (CMA-ES) is a derivative-free method [12] 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

© ICST Institute for Computer Sciences, Social Informatics and Telecommunications Engineering 2018
Y. Chen and T. Q. Duong (Eds.): INISCOM 2017, LNICST 221, pp. 254–265, 2018.
https://doi.org/10.1007/978-3-319-74176-5_22

useful on problems that are non-convex, non-separable, ill-conditioned, multimodal, and with noisy evaluations. Applying CMA-ES requires explicitly a finite-dimensional search space on which solution candidates live. CMA-ES has been used to solve problems in power prediction for smart grid [15], driver assistance [6], smart transportation (which used evolutionary algorithm as a special form of CMA-ES) [2,24], metro regenerative energy [7], etc. In the context of robotics, CMA-ES has been widely used in many tasks: biped locomotion [31], whole-body locomotion optimization [8,9], swimming [26], skill learning via reinforcement learning [13,14,25], inverse reinforcement learning [5,21], 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 [25], policy search [14], or a loss function in the contexts of inverse optimal control [5,21], etc.

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 a RKHS. That means the solution space does not need to depend on any manual parametrization. *Secondly*, by modeling the solution space in RKHS, all updates step 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.

2 Covariance Matrix Adaptation - Evolution Strategy

The Covariance Matrix Adaptation - Evolution Strategy (CMA-ES) is a global optimization method introduced by [12]. 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 kth population of λ candidates from the kth distribution as

Algorithm 1. The CMA-ES algorithm

1: Initialize $\mathbf{m} \in \Re^n$, $\sigma \in \Re^+$, λ, μ
2: Initialize $\mathbf{C} = \mathbf{I}, \mathbf{p}_c = \mathbf{0}, \mathbf{p}_\sigma = \mathbf{0}$
3: **while** (not terminate) **do**
4: Sampling: $\theta_i = \mathbf{m} + \sigma \mathbf{y}_i$, $\mathbf{y}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{C}), i = 1, \ldots, \lambda$
5: Evaluating: $f(\theta_i), i = 1, \ldots, \lambda$
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**

$\theta_i \sim \mathcal{N}(\theta; \mathbf{m}_k, \mathbf{C}_k)$, $i = 1, \cdots, \lambda$. Then, the candidates are sorted ascendingly according to their evaluations $f(\theta_i)$. Only the first μ 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 σ 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 μ candidates as in step 7, in which the weights w_i are set to $1/\mu$ or to a better values $\log(\mu/2) - \log(i)$. 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-μ update which takes into account the *good* variations in the last population. This step-size control update constraints the expected changes of the distribution. Thus, this update in step 10 is based on the *conjugate evolution path* $\mathbf{p}_\sigma$. It targets to accelerate convergence to an optimum, and meanwhile prevents premature convergence. The other parameters: μ_w is the variance effective selection mass, c_1, c_c, c_σ are learning rates, and d_σ is a damping factor for σ. The setting of these parameters is well studied and discussed in-depth in [10].

The updates of CMA-ES can alternatively be derived using the information-geometric concept of a natural gradient as shown in [11], which shares the same insight with the *natural evolution strategies* (NES) [32].

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 [19,20] 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.

3 CMA-ES in Reproducing Kernel Hilbert Space

3.1 Problem Statement

We consider a functional optimization problem [28] 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 μ. 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, denoted as $\mathcal{H}_K$, [17] 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 I_n is an $n \times n$ identity matrix, and κ is a scalar-valued kernel [22]. Each function $h \in \mathcal{H}$ is then represented as a linear span of finite elements $\{x_i, y_i\}$ as

$$h(\cdot) = \sum_i K(x_i, \cdot)y_i \tag{1}$$

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 σ is a scalar global step-size. We discuss now how to update the functionals $\{m, C\}$ and the parameter σ in our CMA-ES-RKHS framework, which is also summarized in Algorithm 2.

Algorithm 2. The CMA-ES-RKHS algorithm

1: Initialize $m \in \mathcal{H}_K$, $\sigma \in \Re^+$, λ, μ
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**

Mean Function Update in RKHS. Assuming that at iteration k, we can sample a set of λ functions $\tilde{g}_i \sim \mathbb{GP}(0, C)$ (Step 4), many sampling techniques are basically described in [18]. Our following derivation is not restricted to which kernel, vector-valued or scalar-valued, is used. A sample from a Gaussian process is not in $\mathcal{H}_K$ with probability of 1, as discussed in detail by [1]. 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),

$$m = m + \sigma \sum_{i=1}^{\mu} w_i g_{1:\lambda} \in \mathcal{H}_K \tag{2}$$

where the 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

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

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 $w_i = \log(\mu + \frac{1}{2}) - \log(i)$. In our experiment, we implement the last choice.

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-μ 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),

$$p_c = (1 - c_c)p_c + \sqrt{c_c(2 - c_c)\mu_w}\bar{g} \in \mathcal{H}_K \tag{3}$$

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-μ is computed as (Step 15),

$$C = (1 - c_\mu - c_1)C + c_1 p_c p_c^\top + c_\mu \sum_{i=1}^{\mu} w_i g_{i:\lambda} \otimes g_{i:\lambda} \tag{4}$$

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

Step-Size Update. The global step-size σ is adapted through the computation of a functional conjugate evolution path as (Step 11),

$$p_\sigma = (1 - c_\sigma)p_\sigma + \sqrt{c_\sigma(2 - c_\sigma)\mu_{\text{eff}}}\mathrm{C}^{-\frac{1}{2}}\bar{g} \tag{5}$$

where c_σ is a backward time horizon for the conjugate evolution path p_σ. According to the bounded inverse theorem in functional analysis [4], C as computed in Eq. 4 is a linear operator in the RKHS $\mathcal{H}_K$, hence it has a bounded inverse C^{-1}. Therefore, p_c 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 σ increased, otherwise decreased. The update formula of σ (Step 12) is

$$\sigma = \sigma \exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{\|p_\sigma\|}{\mathbb{E}\|\mathcal{GP}(0, \delta(\cdot,\cdot))\|} - 1\right)\right) \tag{6}$$

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 $\mathcal{GP}(0, \delta(\cdot,\cdot))$.

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. 5 that involves to find the inverse operator $C^{-\frac{1}{2}}$.

In general, we can use the kernel matching pursuit algorithm [30] 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 [23] for achieving efficiently and fast both a sparse and compact covariance operator and its inverse square root operator.

4 Experiments

We evaluate the advantages and general applications of CMA-ES-RKHS on two optimization problems: a synthetic functional optimization, and a power prediction scenario. 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. 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 the power of our proposed method in capturing a complex solution function which can not be found easily by existing methods, and (iii) the applicability in practice.

4.1 Synthetic Domains

We design an unknown 2-dimensional functions f^*. This function is a mixture of two (multi-variate) Gaussians. 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

$$J(h) = \int_{x_0}^{x_T} \big(f^*(x) - h(x)\big)^2 dx \tag{7}$$

where $x \in \Re^2$. This task is a simplified version of many similar problems in machine learning and robotics, e.g. regularized risk functional [22], trajectory optimization [27,29], trajectory optimization in RKHS [16], loss minimization inverse optimal control [5], 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

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

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 [30] 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, $N = 100$. 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 in Figs. 2 and 3. 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. 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). Figure 2 shows very interesting results where other methods like CMA-ES and CMA-ES-A are still struggling around the optimal regions.

4.2 Power Prediction Scenarios

In this section, we apply CMA-ES-RKHS for prediction of power demand by designing a simulated scenario in which one nation's electricity consumption

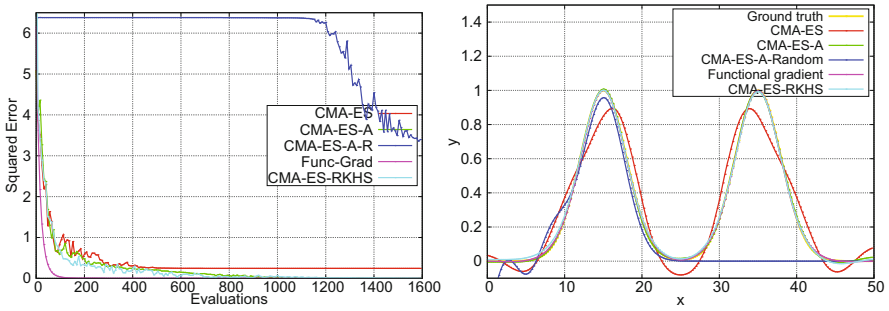

Fig. 1. Power Prediction Scenarios: (left) squared error, (right) solution functions (x-axis: time; y-axis: scaled consumption in mega-watt)

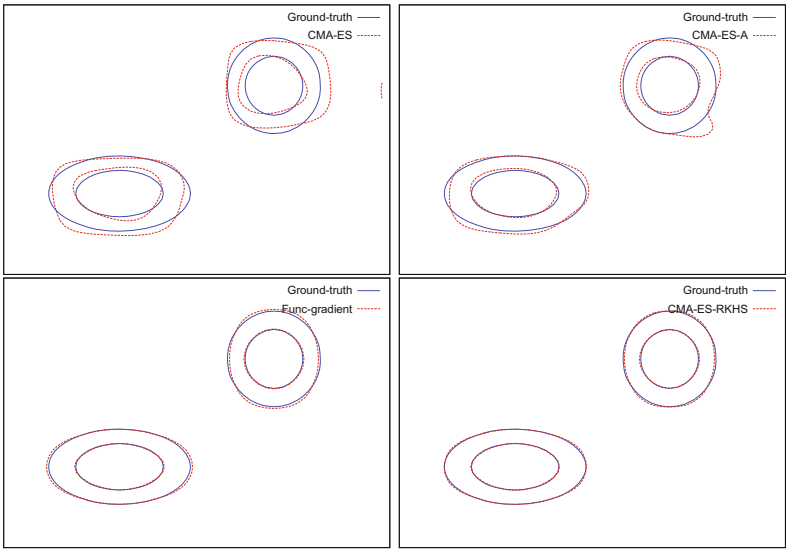

Fig. 2. Solution functions: contours of levels equivalent to the first and second deviations

is given by an unknown function $f^* : \Re \mapsto \Re$ where it maps a moment in continuous time to a real value of mega-watt (MW). The goal is to use a black-box optimization method to estimate that unknown function. For this task, we will also look for an estimate function h that minimizes the least square term J as introduced in Eq. 7. This objective is similar to the work in [15] in which they fit a parametric function over a statistical data.

We also design the unknown consumption function f^* as a mixture of two univariate Gaussians, called the ground-truth function. We use $N = 10$ for this task. We report the squared error J and the solution function in Fig. 1. The performance of CMA-ES-A-R is very bad in terms of error. As demonstrated on the

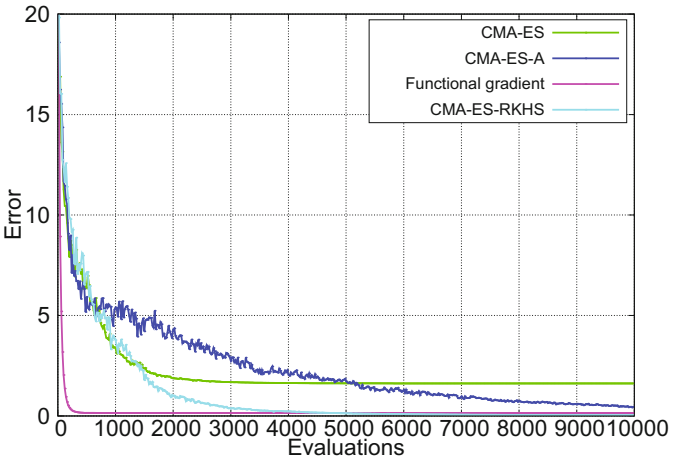

Fig. 3. Results for the 2D synthetic domain

right picture, it can detect only one mode of the optimal function. 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.

5 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. Our experiments show that CMA-ES-RKHS is 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.

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

References

1. Adler, R.J.: The Geometry of Random Fields. Wiley, Chichester (1981)
2. Armas, R., Aguirre, H., Zapotecas-Martínez, S., Tanaka, K.: Traffic signal optimization: minimizing travel time and fuel consumption. In: Bonnevay, S., Legrand, P., Monmarché, N., Lutton, E., Schoenauer, M. (eds.) EA 2015. LNCS, vol. 9554, pp. 29–43. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-31471-6_3
3. Chourabi, H., et al.: Understanding smart cities: an integrative framework. In: 2012 45th Hawaii International Conference on System Science (HICSS), pp. 2289–2297. IEEE (2012)
4. Conway, J.B.: A Course in Functional Analysis, vol. 96. Springer, New York (2013)
5. Doerr, A., Ratliff, N.D., Bohg, J., Toussaint, M., Schaal, S.: Direct loss minimization inverse optimal control. In: Robotics: Science and Systems XI (2015)
6. Edelbrunner, H., Handmann, U., Igel, C., Leefken, I., von Seelen, W.: Application and optimization of neural field dynamics for driver assistance. In: 2001 IEEE Proceedings of the Intelligent Transportation Systems, pp. 309–314. IEEE (2001)
7. Fournier, D., Fages, F., Mulard, D.: A greedy heuristic for optimizing metro regenerative energy usage. In: Railways 2014 (2014)
8. Ha, S., Liu, C.K.: Iterative training of dynamic skills inspired by human coaching techniques. ACM Trans. Graph. **34**(1), 1:1–1:11 (2014)
9. Ha, S., Liu, C.K.: Evolutionary optimization for parameterized whole-body dynamic motor skills. In: ICRA, pp. 1390–1397 (2016)
10. Hansen, N.: The CMA Evolution Strategy: A Tutorial. CoRR, abs/1604.00772 (2016)
11. Hansen, N., Auger, A.: Principled design of continuous stochastic search: from theory to practice. In: Borenstein, Y., Moraglio, A. (eds.) Theory and Principled Methods for the Design of Metaheuristics. NCS, pp. 145–180. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-33206-7_8
12. Hansen, N., Müller, S.D., Koumoutsakos, P.: Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). Evol. Comput. **11**(1), 1–18 (2003)
13. Heidrich-Meisner, V., Igel, C.: Hoeffding and Bernstein races for selecting policies in evolutionary direct policy search. In: ICML, pp. 401–408 (2009)
14. Heidrich-Meisner, V., Igel, C.: Neuroevolution strategies for episodic reinforcement learning. J. Algorithms **64**(4), 152–168 (2009)
15. Kramer, O., Satzger, B., Lässig, J.: Power prediction in smart grids with evolutionary local kernel regression. In: Graña Romay, M., Corchado, E., Garcia Sebastian, M.T. (eds.) HAIS 2010. LNCS (LNAI), vol. 6076, pp. 262–269. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13769-3_32
16. Marinho, Z., Boots, B., Dragan, A., Byravan, A., Gordon, G.J., Srinivasa, S.: Functional gradient motion planning in reproducing kernel Hilbert spaces. In: RSS (2016)
17. Micchelli, C.A., Pontil, M.: On learning vector-valued functions. Neural Comput. **17**(1), 177–204 (2005)
18. Rasmussen, C.E., Williams, C.K.I.: Gaussian Processes for Machine Learning. The MIT Press, Cambridge (2006)
19. Rubinstein, R.: The cross-entropy method for combinatorial and continuous optimization. Methodol. Comput. Appl. Probab. **1**(2), 127–190 (1999)
20. Rubinstein, R.Y., Kroese, D.P.: The cross-entropy method: a unified approach to combinatorial optimization. Monte-Carlo Simulation And Machine Learning. Springer, New York (2013)

21. Rückert, E.A., Neumann, G., Toussaint, M., Maass, W.: Learned graphical models for probabilistic planning provide a new class of movement primitives. Front. Comput. Neurosci. **6**(97), 1–20 (2013)
22. Schölkopf, B., Smola, A.J.: Learning with kernels support vector machines, regularization, optimization, and beyond. Adaptive Computation and Machine Learning series. MIT Press, Cambridge (2002)
23. Schölkopf, B., Smola, A., Müller, K.-R.: Kernel principal component analysis. In: Gerstner, W., Germond, A., Hasler, M., Nicoud, J.-D. (eds.) ICANN 1997. LNCS, vol. 1327, pp. 583–588. Springer, Heidelberg (1997). https://doi.org/10.1007/BFb0020217
24. Stolfi, D.H., Alba. E.: Eco-friendly reduction of travel times in European smart cities. In: Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation, pp. 1207–1214. ACM (2014)
25. Stulp, F., Sigaud, O.: Path integral policy improvement with covariance matrix adaptation. In: ICML (2012)
26. Tan, J., Gu, Y., Turk, G., Liu, C.K.: Articulated swimming creatures. ACM Trans. Graph. (TOG) **30**(4), 58 (2011)
27. Toussaint, M.: Newton Methods for K-order Markov Constrained Motion Problems. CoRR, abs/1407.0414 (2014)
28. Ulbrich, M.: Optimization with PDE Constraints. Optimization methods in Banach spaces, pp. 97–156. Springer, Dordrecht (2009)
29. Vien, N.A., Toussaint, M.: POMDP manipulation via trajectory optimization. In: 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2015, September 28 - October 2, Hamburg, pp. 242–249 (2015)
30. Vincent, P., Bengio, Y.: Kernel matching pursuit. Mach. Learn. **48**(1–3), 165–187 (2002)
31. Wang, J.M., Hamner, S.R., Delp, S.L., Koltun, V.: Optimizing locomotion controllers using biologically-based actuators and objectives. ACM Trans. Graph **31**(4), 25:1–25:11 (2012)
32. Wierstra, D., Schaul, T., Glasmachers, T., Sun, Y., Peters, J., Schmidhuber, J.: Natural evolution strategies. J. Mach. Learn. Res. **15**(1), 949–980 (2014)