Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16)

Derivative-Free Optimization via Classification*

Yang Yu and **Hong Qian** and **Yi-Qi Hu**
National Key Laboratory for Novel Software Technology,
Nanjing University, Nanjing 210023, China
{yuy,qianh,huyq}@lamda.nju.edu.cn

Abstract

Many randomized heuristic derivative-free optimization methods share a framework that iteratively learns a model for promising search areas and samples solutions from the model. This paper studies a particular setting of such framework, where the model is implemented by a classification model discriminating good solutions from bad ones. This setting allows a general theoretical characterization, where critical factors to the optimization are discovered. We also prove that optimization problems with *Local Lipschitz* continuity can be solved in polynomial time by proper configurations of this framework. Following the critical factors, we propose the *randomized coordinate shrinking* classification algorithm to learn the model, forming the RACOS algorithm, for optimization in continuous and discrete domains. Experiments on the testing functions as well as on the machine learning tasks including spectral clustering and classification with Ramp loss demonstrate the effectiveness of RACOS.

Introduction

Sophisticated optimization problems are often encountered in numerous real-world applications and are in the core of many fields. In the realm of optimization, these problems can be formalized as $\mathrm{argmin}_{x \in X} f(x)$, where X is the domain. In many optimization tasks, we cannot assume that f has good properties such as linearity, convexity, or even differentiability. Thus derivative-free optimization methods are quite appealing, including genetic algorithms (Golberg 1989), randomized local search (Neumann and Wegener 2007), estimation of distribution algorithms (Larrañaga and Lozano 2002), cross-entropy methods (de Boer et al. 2005), Bayesian optimization methods (Brochu, Cora, and De Freitas 2010), optimistic optimization methods (Munos 2014), etc. However, due to the extremely large variety of the problems and the heuristics in the algorithms, most derivative-free optimization algorithms are very weak in their theoretical foundations, only except the recent development of the optimistic optimization method (Munos 2011; 2014), Bayesian optimization methods (Bull 2011; de Freitas, Smola, and Zoghi 2012), and a few evolutionary algorithms (Yu, Yao, and Zhou 2012; Qian, Yu, and Zhou 2015a; 2015b).

Many derivative-free optimization methods are model-based, i.e., they learn a model from the solutions evaluated in the optimization path, and the model is then used to guide the sampling of solutions for the next round. For example, cross-entropy methods may use Gaussian distribution as the model, Bayesian optimization methods employ Gaussian process to model the joint distribution, and the estimation of distribution algorithms have incorporated many kinds of learning models.

In this paper we study model-based optimization methods with a particular type of model: classification model. A classification model learns to classify solutions into two categories, *good* or *bad*. The learned model partitions the solution space into *good* and *bad* areas. Then solutions are sampled from the *good* areas. Employing a classification model is not a new idea, since many different learning models have been explored (e.g., (Lozano et al. 2006)). However, the theoretical understanding was only roughly touched (Yu and Qian 2014). Important questions need to be addressed, including which factors of a classification model effect the optimization performance, what problems can be efficiently solved. Towards this direction, this paper makes two contributions:

• We prove a general optimization performance bound of a classification-based optimization framework, which directly depends on two critical factors of the classification models: the *error-target dependence* and the *shrinking rate*. We also show that problems with *Local Lipschitz* continuity can be solved in polynomial time, if the framework is properly configured.

• Following the critical factors, we design the *randomized coordinate shrinking* classification algorithm, which learns to discriminate good and bad solutions while trying to keep the the error-target dependence and the shrinking rate small. Employing this algorithm, we propose the RACOS optimization algorithm for both continuous and discrete domains. We then conduct experiments comparing RACOS with some state-of-the-art derivative-free optimization methods on optimization testing functions and machine learning tasks including spectral clustering and classification with Ramp loss. The experiment results show that RACOS is superior

*This research was supported by the NSFC (61375061, 61272217, 61223003), Foundation for the Author of National Excellent Doctoral Dissertation of China (201451).

to the compared methods.

The rest of this paper is organized in five sections, sequentially presenting the background, the theoretical study, the proposed RACOS algorithm, the empirical results, and the conclusion.

Background

This paper considers the general minimization problems in continuous and finite discrete domains. Let X denote a solution space that is a compact subset of $\mathbb{R}^n$, and $f\colon X \to \mathbb{R}$ denote a minimization problem. Assume that there exist $x^*, x' \in X$ such that $f(x^*) = \min_{x\in X} f(x)$ and $f(x') = \max_{x\in X} f(x)$. Let $\mathcal{F}$ denote the set of all functions that satisfy this assumption. Given $f \in \mathcal{F}$, the *minimization problem* is to find a solution $x^* \in X$ s.t. $\forall x \in X : f(x^*) \leq f(x)$. For derivative-free optimization algorithms, we assume that only an objective function $f \in \mathcal{F}$ is accessible to the algorithm for evaluating solutions. Other information of f such as derivatives should be unknown to the algorithm.

For the purpose of theoretical analysis, given $f \in \mathcal{F}$, let $\underline{M}_f$ and $\overline{M}_f$ be its known lower and upper bounds, respectively. For a subset $D \subseteq X$, let $\#D = \int_{x\in X} \mathbb{I}[x \in D]\,\mathrm{d}x$ (or $\#D = \sum_{x\in X} \mathbb{I}[x \in D]$ for finite discrete domains), where $\mathbb{I}[\cdot]$ is the indicator function. Define $|D| = \#D/\#X$ and thus $|D| \in [0, 1]$. Let $D_\alpha = \{x \in X \,|\, f(x) \leq \alpha\}$ for $\alpha \in [\underline{M}_f, \overline{M}_f]$, and $D_\epsilon = \{x \in X \,|\, f(x) - f(x^*) \leq \epsilon\}$ for $\epsilon > 0$.

A hypothesis (or a classifier) h is a function mapping the solution space X to $\{-1, +1\}$. Let $\mathcal{H} \subseteq \{h\colon X \to \{-1, +1\}\}$ be a hypothesis space consisting of candidate hypotheses h. Let $D_h = \{x \in X \,|\, h(x) = +1\}$ for hypothesis $h \in \mathcal{H}$, i.e., the positive class region represented by h. Denote $\mathcal{U}_X$ and $\mathcal{U}_{D_h}$ the uniform distribution over X and D_h, respectively, and denote $\mathcal{T}_h$ the distribution defined on D_h induced by h. Let $\mathrm{sign}[v]$ be the sign function returning 1 if $v \geq 0$ and -1 otherwise.

A simplified classification-based optimization (Yu and Qian 2014) is presented in Algorithm 1. It starts from a set of uniformly generated solutions (line 1), and then a cycle (lines 3 to 11) is followed. In each iteration, the algorithm queries the objective function to assess the generated solutions, and forms a binary classification data set B_t (line 4), where a threshold α_t is used to label the solutions as positive and negative according to $\mathrm{sign}[\alpha_t - f(x)]$. In the classification phase (line 7), a binary classifier is trained on B_t, in order to approximate the region $D_{\alpha_t} = \{x \in X \,|\, f(x) \leq \alpha_t\}$. During the sampling step (line 8), solutions are generated by invoking the `Sampling` sub-procedure. We specify the `Sampling`(h, λ) as that, it samples with probability λ from $\mathcal{U}_{D_h}$ (the uniform distribution over the positive region classified by h), and with the remaining probability from $\mathcal{U}_X$ (the uniform distribution over X). Throughout the procedure, the best-so-far solutions are recorded (line 1 and line 11), and the best one will be returned as the output solution (line 12). Note that an efficient sampling from arbitrary region is nontrivial, such as the sampling in the Bayesian optimization methods (Brochu, Cora, and De Freitas 2010). In our latter proposed learning algorithm, the region will be a rectangle, and thus the sampling is straightforward.

Algorithm 1 classification-based optimization

Input:
- f: Objective function to be minimized;
- $\mathcal{C}$: A binary classification algorithm;
- $\lambda \in [0, 1]$: Balancing parameter;
- $\alpha_1 > \ldots > \alpha_T$: Threshold for labeling;
- $T \in \mathbb{N}^+$: Number of iterations;
- $m \in \mathbb{N}^+$: Sample size in each iteration;
- `Sampling`: Sampling sub-procedure.

Procedure:

1: Collect $S_0 = \{x_1, \ldots, x_m\}$ by i.i.d. sampling from $\mathcal{U}_X$
2: Let $\tilde{x} = \mathrm{argmin}_{x\in S_0} f(x)$
3: **for** $t = 1$ to T **do**
4: Construct $B_t = \{(x_1, y_1), \ldots, (x_m, y_m)\}$, where $x_i \in S_{t-1}$ and $y_i = \mathrm{sign}[\alpha_t - f(x_i)]$
5: Let $S_t = \emptyset$
6: **for** $i = 1$ to m **do**
7: $h_t = \mathcal{C}(B_t)$, where $h_t \in \mathcal{H}$
8: $x_i = \texttt{Sampling}(h_t, \lambda)$, and let $S_t = S_t \cup \{x_i\}$
9: **end for**
10: $\tilde{x} = \mathrm{argmin}_{x\in S_t\cup\{\tilde{x}\}} f(x)$
11: **end for**
12: **return** $\tilde{x}$ and $f(\tilde{x})$

The simplicity of classification-based optimization admits a general performance bound on the query complexity (Definition 1) (Yu and Qian 2014), which counts the total number of calls to the objective function by an algorithm $\mathcal{A}$ before it finds a solution that reaches the approximation level ϵ, with high probability.

DEFINITION 1 ((ϵ,δ)-Query Complexity)
Given $f \in \mathcal{F}$, an algorithm $\mathcal{A}$, $0 < \delta < 1$ and $\epsilon > 0$, the (ϵ,δ)-query complexity is the number of calls to f such that, with probability at least $1 - \delta$, $\mathcal{A}$ finds at least one solution $\tilde{x} \in X \subseteq \mathbb{R}^n$ satisfying

$$f(\tilde{x}) - f(x^*) \leq \epsilon,$$

where $f(x^*) = \min_{x\in X} f(x)$.

Lemma 1 and Lemma 2 are simplified from (Yu and Qian 2014). In order to be self-contained, their proofs are included in the appendix[1]. For the lemmas, let $\mathcal{D}_t = \lambda\mathcal{U}_{D_{h_t}} + (1 - \lambda)\mathcal{U}_X$ be the sampling distribution at iteration t, $R_{\mathcal{D}_t}$ denote the generalization error of $h_t \in \mathcal{H}$ with respect to the target function under distribution $\mathcal{D}_t$, and D_{KL} denote the Kullback-Leibler (KL) divergence between two probability distributions. Combining the two lemmas results in a general and explicit query complexity upper bound of any algorithm following the classification-based optimization framework for any $f \in \mathcal{F}$.

LEMMA 1
Given $f \in \mathcal{F}$, $0 < \delta < 1$ and $\epsilon > 0$, the (ϵ,δ)-query complexity of a classification-based optimization algorithm is

[1]The appendix presenting all the proofs can be found in the webpage of the author http://cs.nju.edu.cn/yuy

upper bounded by

$$O\left(\max\left\{\frac{1}{(1-\lambda)|D_\epsilon|+\lambda\overline{\mathbf{Pr}}_h}\ln\frac{1}{\delta},\sum_{t=1}^{T}m_{\mathbf{Pr}_{h_t}}\right\}\right),$$

where $\overline{\mathbf{Pr}}_h = \frac{1}{T}\sum_{t=1}^{T}\mathbf{Pr}_{h_t} = \frac{1}{T}\sum_{t=1}^{T}\int_{D_\epsilon}\mathcal{U}_{D_{h_t}}(x)\,\mathrm{d}x$ *(or* $\overline{\mathbf{Pr}}_h = \frac{1}{T}\sum_{t=1}^{T}\sum_{x\in D_\epsilon}\mathcal{U}_{D_{h_t}}(x)$ *for finite discrete domains) is the average success probability of sampling from the learned positive region of* h_t*, and* $m_{\mathbf{Pr}_{h_t}}$ *is the sample size required to realize the success probability* $\mathbf{Pr}_{h_t}$*.*

LEMMA 2
Given $f\in\mathcal{F}$*,* $\epsilon>0$*, the average success probability of sampling from the distributions induced by the learned hypotheses of any classification-based optimization algorithm* $\overline{\mathbf{Pr}}_h$ *is lower bounded by*

$$\overline{\mathbf{Pr}}_h \geq \frac{1}{T}\sum_{t=1}^{T}\frac{|D_\epsilon|-2\Psi^{R_{\mathcal{D}_t}}_{D_{KL}(\mathcal{D}_t\|\mathcal{U}_X)}}{|D_{\alpha_t}|+\Psi^{R_{\mathcal{D}_t}}_{D_{KL}(\mathcal{D}_t\|\mathcal{U}_X)}},$$

where $\Psi^{R_{\mathcal{D}_t}}_{D_{KL}(\mathcal{D}_t\|\mathcal{U}_X)} = R_{\mathcal{D}_t}+\#X\sqrt{\frac{1}{2}D_{KL}(\mathcal{D}_t\|\mathcal{U}_X)}$.

Let $\mathtt{poly}(\cdot)$ be the set of all polynomials w.r.t. the related variables and $\mathtt{superpoly}(\cdot)$ be the set of all functions that grow faster than any function in $\mathtt{poly}(\cdot)$ w.r.t. the related variables.

Theoretical Study

Lemma 1 indicates that classification-based optimization algorithms will degenerate to the uniformly random search if $\lambda=0$. Now, we only consider that $0<\lambda\leq 1$. Since $\overline{\mathbf{Pr}}_h$ is related to the learned hypothesis and the classification phase can be arbitrarily bad without any restriction, a classification-based optimization algorithm does not always outperform the uniformly random search. In the worst case, $\overline{\mathbf{Pr}}_h$ can be zero and thus the classification-based optimization algorithm degenerates to the uniformly random search, which can be derived directly from Lemma 1 and Lemma 2. However, as long as $\overline{\mathbf{Pr}}_h$ is larger than $|D_\epsilon|$, the part of the local search dominates the query complexity, and the corresponding classification-based optimization algorithm outperforms the uniformly random search. Furthermore, we hope that, given $f\in\mathcal{F}'\subseteq\mathcal{F}$, the query complexity of the corresponding classification-based optimization algorithm belongs to $\mathtt{poly}(\frac{1}{\epsilon},\frac{1}{\delta},n)$, i.e., f is efficiently approximative by classification-based optimization algorithms. Therefore, we are interested in investigating the conditions under which f is efficiently approximative by classification-based optimization algorithms.

Definition 2 characterizes the dependence between the classification error and the target region. The smaller θ indicates that they are more independent, and when $\theta=0$, they are totally independent. We would want the classification error and the target region to be independent, such that sampling from the positive region of the classifier may obtain solutions in the target region.

DEFINITION 2 (Error-Target θ-Dependence)
The error-target dependence $\theta\geq 0$ *of a classification-based optimization algorithm is its infimum such that, for any* $\epsilon>0$ *and any* t*,*

$$\begin{aligned}|D_\epsilon|\cdot|D_{\alpha_t}\Delta D_{h_t}|-\theta|D_\epsilon| &\leq |D_\epsilon\cap(D_{\alpha_t}\Delta D_{h_t})|\\ &\leq |D_\epsilon|\cdot|D_{\alpha_t}\Delta D_{h_t}|+\theta|D_\epsilon|,\end{aligned}$$

where the operator Δ *is the symmetric difference of two sets defined as* $A_1\Delta A_2=(A_1\cup A_2)-(A_1\cap A_2)$*. It characterizes, when sampling a solution* x *from* $\mathcal{U}_X$*, the dependence between the random variable that whether* $x\in D_{\alpha_t}\Delta D_{h_t}$ *and the random variable that whether* $x\in D_\epsilon$*.*

Definition 3 characterizes how large the positive region of the classifier. The smaller γ indicates the smaller the positive region. When the dependence between the classification error and the target region is low, we would want the positive region to be small, so that the chance of sampling within the target region could be high.

DEFINITION 3 (γ-Shrinking Rate)
The shrinking rate $\gamma>0$ *of a classification-based optimization algorithm is its infimum such that* $|D_{h_t}|\leq\gamma|D_{\alpha_t}|$ *for all* t*.*

Using these two definitions, we present a general upper bound of the query complexity of a classification-based optimization algorithm.

THEOREM 1
Given $f\in\mathcal{F}$*,* $0<\delta<1$ *and* $\epsilon>0$*, if a classification-based optimization algorithm has error-target* θ*-dependence and* γ*-shrinking rate, its* (ϵ,δ)*-query complexity is upper bounded*

$$O\left(\frac{1}{|D_\epsilon|}\left((1-\lambda)+\frac{\lambda}{\gamma T}\sum_{t=1}^{T}\frac{1-Q\cdot R_{\mathcal{D}_t}-\theta}{|D_{\alpha_t}|}\right)^{-1}\ln\frac{1}{\delta}\right),$$

where $Q=1/(1-\lambda)$*.*

The proof is presented in the appendix. Theorem 1 discloses that the error-target θ-dependence and the γ-shrinking rate are two important factors. It can be observed that the smaller θ and γ, the better the query complexity.

On the basis of Lemma 1 and Theorem 1, we then study what function classes can be efficiently optimized by classification-based optimization algorithms.

First, we find that a class of functions $\mathcal{F}_L\subseteq\mathcal{F}$ satisfying the *Local Lipschitz* continuity (Definition 4) can be efficiently optimized by classification-based optimization algorithms with error-target dependence $\theta<1$ and shrinking rate $\gamma>0$. For finite discrete domains, we consider $X=\{0,1\}^n$ and let $\|x-y\|_H$ denote the Hamming distance between $x,y\in\{0,1\}^n$.

DEFINITION 4 (Local Lipschitz)
Given $f\in\mathcal{F}$*, let* x^* *be a global minimum of* f*, for all* $x\in X$*, if* $X=\{0,1\}^n$*, then there exist positive constants* β_1,β_2,L_1,L_2 *such that*

$$L_2\|x-x^*\|_H^{\beta_2}\leq f(x)-f(x^*)\leq L_1\|x-x^*\|_H^{\beta_1};$$

if X is a compact continuous domains, then there exist positive constants $\beta_1, \beta_2, L_1, L_2$ such that

$$L_2\|x - x^*\|_2^{\beta_2} \leq f(x) - f(x^*) \leq L_1\|x - x^*\|_2^{\beta_1}.$$

Let $\mathcal{F}_L^{\beta_1,L_1,\beta_2,L_2} (\subseteq \mathcal{F})$ denote the function class that satisfies the condition.

This condition guarantees that f has a bounded change range around a global minimum x^*. Note that we can have classification algorithms with the convergence rate of the generalization error $\widetilde{O}(\frac{1}{m})$ ignoring other variables and logarithmic terms (Kearns and Vazirani 1994; Vapnik 2000), where m is the sample size for the learning. Thus we assume that the classification-based optimization algorithms use the classification algorithms with convergence rate $\widetilde{\Theta}(\frac{1}{m})$.

COROLLARY 1
In finite discrete domains $X = \{0,1\}^n$, given $f \in \mathcal{F}_L^{\beta_1,L_1,\beta_2,L_2}$, $0 < \delta < 1$ and $0 < \epsilon \leq L_1(\frac{n}{2})^{\beta_1}$, for a classification-based optimization algorithm using a classification algorithm with convergence rate $\widetilde{\Theta}(\frac{1}{m})$, under the conditions that error-target dependence $\theta < 1$ and shrinking rate $\gamma > 0$, its (ϵ,δ)-query complexity belongs to $poly(\frac{1}{\epsilon}, n, \frac{1}{\beta_1}, \beta_2, \ln L_1, \ln \frac{1}{L_2}) \cdot \ln \frac{1}{\delta}$.

COROLLARY 2
In compact continuous domains X, given $f \in \mathcal{F}_L^{\beta_1,L_1,\beta_2,L_2}$, $0 < \delta < 1$ and $\epsilon > 0$, for a classification-based optimization algorithm using a classification algorithm with convergence rate $\widetilde{\Theta}(\frac{1}{m})$, under the conditions that error-target dependence $\theta < 1$ and shrinking rate $\gamma > 0$, its (ϵ,δ)-query complexity belongs to $poly(\frac{1}{\epsilon}, n, \frac{1}{\beta_1}, \beta_2, \ln L_1, \ln \frac{1}{L_2}) \cdot \ln \frac{1}{\delta}$.

More generally, instead of the *Local Lipschitz* continuity for compact continuous domains, we present another sufficient condition under which f can be efficiently optimized by classification-based optimization algorithms, using the η-Packing Number and η-Covering Number (Definition 5). Recall that $D_\epsilon = \{x \in X \mid f(x) - f(x^*) \leq \epsilon\}$ for any $\epsilon > 0$. Let $\alpha_t' = \alpha_t - f(x^*)$ and we assume that $\alpha_t' > 0$.

DEFINITION 5 (η-Packing Number and η-Covering Number)
η-Packing Number is the largest $\mathcal{N}_p \geq 0$ such that, there exists $C_1 > 0$, for all $\epsilon > 0$, the maximal number of disjoint ℓ_2-balls of radius $\eta\epsilon$ contained in D_ϵ with center in D_ϵ is not less than $C_1\epsilon^{-\mathcal{N}_p}$; η-Covering Number is the smallest $\mathcal{N}_c \geq 0$ such that, there exists $C_2 > 0$, for all $\epsilon > 0$, the minimal number of ℓ_2-balls of radius $\eta\epsilon$ with center in X covering D_ϵ is not larger than $C_2\epsilon^{-\mathcal{N}_c}$.

COROLLARY 3
In compact continuous domains X, given $f \in \mathcal{F}$ satisfying $\sum_{t=1}^{T} (\alpha_t')^{\mathcal{N}_c - n} \in \Omega\left(\epsilon^{\mathcal{N}_p - n}\right)$, where $\mathcal{N}_p$ and $\mathcal{N}_c$ are its η-Packing and η-Covering numbers, respectively, $0 < \delta < 1$ and $\epsilon > 0$, for a classification-based optimization algorithm using the classification algorithms with convergence rate $\widetilde{\Theta}(\frac{1}{m})$, under the conditions that error-target dependence $\theta < 1$ and shrinking rate $\gamma > 0$, its (ϵ,δ)-query complexity belongs to $poly(\frac{1}{\epsilon}, n) \cdot \ln \frac{1}{\delta}$.

The proofs of the corollaries are in the appendix. For continuous domains, we have $\mathcal{N}_p \leq \mathcal{N}_c$. Because if we let $V(D_\epsilon)$ and $V(\eta\epsilon)$ denote the volume of D_ϵ and ℓ_2 ball of radius $\eta\epsilon$ in $\mathbb{R}^n$ respectively, then it holds that $C_1\epsilon^{-\mathcal{N}_p} \cdot V(\eta\epsilon) \leq V(D_\epsilon) \leq C_2\epsilon^{-\mathcal{N}_c} \cdot V(\eta\epsilon)$. It is worthwhile to point out that if $\mathcal{N}_c = \mathcal{N}_p = n$, the condition $\sum_{t=1}^{T} (\alpha_t')^{\mathcal{N}_c - n} \in \Omega\left(\epsilon^{\mathcal{N}_p - n}\right)$ can always be satisfied, which implies that classification-based optimization is efficient on this class of functions.

The RACOS Algorithm

Theorem 1 has revealed that two critical factors, the error-target dependence and shrinking rate, should be as small as possible. Notice that these two factors were not in traditional classification algorithms. Thus, we need to design new classification algorithm. Inspired by the classical and simple *version space* learning algorithm (Mitchell 1997), we propose the *randomized coordinate shrinking* classification algorithm. Given a set of positive and negative solutions, it maintains an axis-parallel rectangle to cover all the positive but no negative solutions, meanwhile the learning is highly randomized and the rectangle is largely shrunk to meet the critical factors.

The detail of the proposed learning algorithm is depicted in Algorithm 2. In each run, it is inputed a set of solutions with their objective values in B_t, which consists of positive and negative solutions according to the threshold α_t (in Algorithm 1). The algorithm discriminates a randomly selected positive solution (line 3) from the negative ones. In line 4, recall that D_{h_t} denotes the positive region of h_t, and I is the index set of dimensions.

The algorithm takes two steps: learning with randomness until all negative solutions have been excluded (lines 5-21) and shrinking (lines 22-25). In learning, we consider $X = \{0,1\}^n$ and $X = [0,1]^n$ for discrete domain (lines 6-9) and continuous domain (lines 10-20), respectively. This can be extended to larger vocabulary sets or general box constraints directly. For discrete domain, it randomly selects a dimension and collapses the dimension to the value of the positive solution (lines 7-8); for continuous domain, it sets the upper or lower bound on a randomly chosen dimension to exclude negative solutions (lines 13-19). Finally, lines 22-25 further shrink the classifier to leave only M dimensions uncollapsed, for both discrete and continuous domains. This learning algorithm with high-level randomness achieves a positive region with a small error-target dependence, and largely reduces the positive region for a small shrinking rate.

By equipping this classification algorithm into Algorithm 1 (implementing the $\mathcal{C}$ in line 7), we obtain the RACOS optimization algorithm. Notice that the `Sampling` procedure of Algorithm 1 (line 8) simply draws a solution from the rectangle positive area uniformly. The codes of RACOS can be found from http://cs.nju.edu.cn/yuy.

Experiments

We use the same fixed parameters for RACOS in all the following experiments: in Algorithm 1 we set $\lambda = 0.95$,

Table 1: Comparing the achieved objective values of the algorithms (mean ± standard derivation). In each column, an entry is bolded if its mean value is the best (smallest); and an entry is marked with bullet if it is significantly worse than the best algorithm by t-test with confidence level 5%. The last column counts the win/tie/loss of the algorithm to RACOS.

Algorithm	*Sonar*	*Heart*	*Ionosphere*	*Breast Cancer*	*German*	w/t/l to RACOS
USC	3.91±0.00•	79.67±0.00•	54.21±0.00•	200.62±0.00•	239.00±0.00•	0 / 0 / 5
GA	3.14±0.74	**57.31**±0.46	55.71±3.74•	189.52±1.26	205.61±1.80•	0 / 3 / 2
RLS	4.07±0.82•	58.81±0.45•	58.74±2.81•	192.63±1.62•	207.36±2.11•	0 / 0 / 5
UMDA	7.40±2.26•	58.76±1.02•	61.77±4.54•	193.58±3.56•	212.83±1.08•	0 / 0 / 5
CE	8.00±1.35•	58.75±1.39•	63.71±3.41•	188.76±3.77	209.57±1.96•	0 / 1 / 4
RACOS	**2.88**±0.63	57.45±0.89	**50.01**±2.80	**187.55**±3.01	**192.11**±2.51	- / - / -

$m = 100$, and α_t is set so that only the best solution is positive, and in Algorithm 2 we set $M = 1$.

On Testing Functions

We first empirically test RACOS on two benchmark testing functions: the convex Sphere function defined as

$$f(x) = \sum_{i=1}^{n} (x_i - 0.2)^2,$$

and the highly non-convex Ackley function defined as

$$f(x) = -20e^{\left(-\frac{1}{5}\sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i-0.2)^2}\right)} - e^{\left(\frac{1}{n}\sum_{i=1}^{n}\cos 2\pi x_i\right)} + e + 20.$$

The functions are minimized within the solution space $X = [0, 1]^n$, of which the minimum values are 0.

RACOS is compared with simultaneous optimistic optimization (SOO) algorithm (Munos 2011; 2014), random embedding Bayesian optimization (REMBO) algorithm (Wang et al. 2013), and covariance matrix adaptation evolution strategy (CMA-ES) algorithm (Hansen, Müller, and Koumoutsakos 2003), where the implementations are by their authors. To study the scalability w.r.t. the solution space dimensions n, we choose n be to $10, 100, 500, 1000$, and set the maximum number of function evaluations to be $30n$ for all algorithms. To study the convergence rate w.r.t. the number of function evaluations, we choose $n = 500$, and set the total number of function evaluations from 5×10^3 to 2×10^5 for the Sphere function and 5×10^3 to 5×10^4 for the Ackley function. Each algorithm is repeated 30 times independently. The mean of the achieved objective values are shown in Figure 1.

Figure 1 (a) and (b) show that RACOS has the lowest growing rate as the dimension increases, indicating that RACOS has a better scalability than the compared algorithms; (c) and (d) show that RACOS reduces the objective function value with the highest rate, indicating that it converges consistently faster than the others.

On Clustering

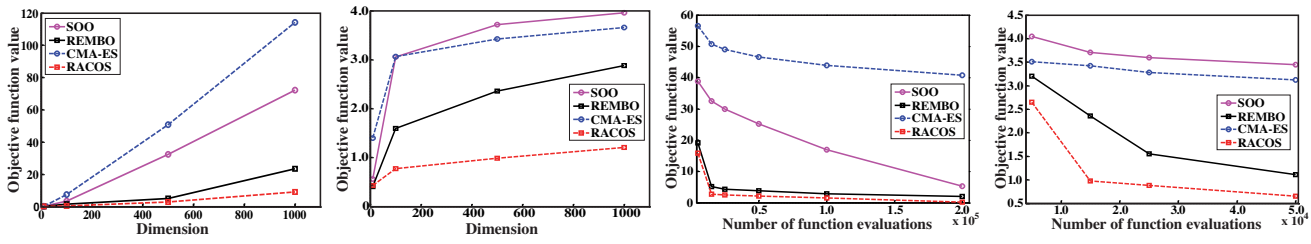
We then study on a clustering task: consider clustering a dataset $\mathcal{V} = \{v_1, \ldots, v_n\}$ into two groups, $\{A_1, A_2\}$. A nature solution space is the discrete domain $X = \{0, 1\}^n$ for the bipartition. The optimization task is to minimize inter-cluster similarity:

$$f(A_1, A_2) = \sum_{i}^{2} \frac{1}{\#A_i} \sum_{p \in A_i, q \notin A_i} W_{p,q} \text{ over } X,$$

where $W_{p,q} = \exp\left(-\|v_p - v_q\|_2^2/\sigma^2\right)$ is the similarity between v_p and v_q. This is also known as the RatioCut problem, and it is NP-hard.

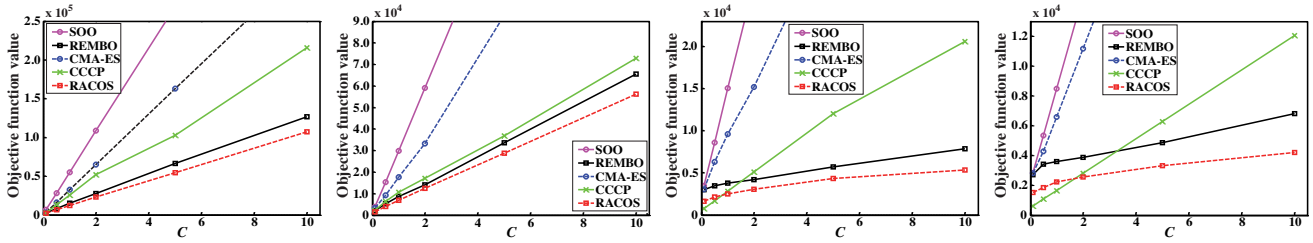
RACOS is compared with unnormalized spectral clustering (USC) algorithm (Von Luxburg 2007) which is a classical approximate algorithm for the RatioCut problem, genetic algorithm (GA) (Golberg 1989) (using the bit-wise mutation with probability $1/n$ and one-bit crossover with probability 0.5), randomized local search (RLS) (Neumann and Wegener 2007), univariate marginal distribution algorithm (UMDA) (Mühlenbein 1997) and cross-entropy (CE) method (de Boer et al. 2005) with the recommended parameters in their references. Five binary UCI datasets (Blake, Keogh, and Merz 1998) are employed: *Sonar*, *Heart*, *Ionosphere*, *Breast Cancer* and *German*, with 208, 270, 351, 683 and 1000 instances, respectively. All features are normalized into $[-1, 1]$. We set the total number of calls to the objective function of GA, RLS, UMDA, CE and RACOS to be $30n$. Each algorithm is repeated 30 times independently on each dataset. Table 1 reports the achieved objective values.

Table 1 shows that, by t-test with confidence level 5%, RACOS is never worse than the others, is always better than



(a) on Sphere function (b) on Ackley function (c) on Sphere function (d) on Ackley function

Figure 1: Comparing the scalability with $30n$ evaluations in (a) & (b), and the convergence rate with $n = 500$ in (c) & (d).



(a) on *Adult*, $s = -1$ (b) on *Adult*, $s = 0$ (c) on *USPS+N*, $s = -1$ (d) on *USPS+N*, $s = 0$

Figure 2: Comparing the achieved objective function values against the parameter C of the classification with Ramp loss.

Algorithm 2 The *randomized coordinate shrinking* classification algorithm for $X = \{0,1\}^n$ or $[0,1]^n$

Input:

t: Current iteration number;
B_t: Solution set in iteration t;
X: Solution space ($\{0,1\}^n$ or $[0,1]^n$);
I: Index set of coordinates;
$M \in \mathbb{N}^+$: Maximum number of uncertain coordinates.

Procedure:

$B_t^+ =$ the positive solutions in B_t
$B_t^- = B_t - B_t^+$
Randomly select $x_+ = (x_+^{(1)}, \ldots, x_+^{(n)})$ from B_t^+
Let $D_{h_t} = X$, $I = \{1, \ldots, n\}$
while $\exists x \in B_t^-$ s.t. $h_t(x) = +1$ **do**
  if $X = \{0,1\}^n$ **then**
    $k =$ randomly selected index from the index set I
    $D_{h_t} = D_{h_t} - \{x \in X \mid x^{(k)} \neq x_+^{(k)}\}$, $I = I - \{k\}$
  end if
  if $X = [0,1]^n$ **then**
    $k =$ randomly selected index from the index set I
    $x^- =$ randomly selected solution from B_t^-
    if $x_+^{(k)} \geq x_-^{(k)}$ **then**
      $r =$ uniformly sampled value in $(x_-^{(k)}, x_+^{(k)})$
      $D_{h_t} = D_{h_t} - \{x \in X \mid x^{(k)} < r\}$
    else
      $r =$ uniformly sampled value in $(x_+^{(k)}, x_-^{(k)})$
      $D_{h_t} = D_{h_t} - \{x \in X \mid x^{(k)} > r\}$
    end if
  end if
end while
while $\#I > M$ **do**
  $k =$ randomly selected index from the index set I
  $D_{h_t} = D_{h_t} - \{x \in X \mid x^{(k)} \neq x_+^{(k)}\}$, $I = I - \{k\}$
end while
return h_t

USC, RLS, UMDA, and have significant wins to GA and CE. The results imply that the performance of RACOS is not only superior to the compared methods, but also stable.

On Classification with Ramp Loss

We finally study on a classification task with Ramp loss (Collobert et al. 2006). The Ramp loss is defined as $R_s(z) = H_1(z) - H_s(z)$ with $s < 1$, where $H_s(z) = \max\{0, s - z\}$ is the Hinge loss with s being the Hinge point. The task is to find a vector w and a scalar b to minimize

$$f(w,b) = \tfrac{1}{2}\|w\|_2^2 + C\sum_\ell^L R_s\big(y_\ell(w^\top v_\ell + b)\big),$$

where v_ℓ is the training instance and $y_\ell \in \{-1, +1\}$ is its label. This objective function is similar to that of support vector machines (SVM) (Vapnik 2000) but the loss function of SVM is the Hinge loss. Due to the convexity of the Hinge loss, the number of support vectors increases linearly with the number of training instances in SVM, which is undesired with respect to scalability. While this problem can be relieved by using the Ramp loss (Collobert et al. 2006).

RACOS is compared with SOO, REMBO, CMA-ES, and the concave-convex procedure (CCCP) (Yuille and Rangarajan 2001) which is a gradient-based non-convex optimization approach for objective functions that can be decomposed into convex sub-function plus concave sub-function. We employ two binary class UCI datasets, *Adult* and *USPS+N* (0 v.s. rest), that are used in (Collobert et al. 2006). The feature dimension of which are 123 and 256, respectively. All features are normalized into $[0,1]$ or $[-1,1]$. Since we focus on the optimization performance, we compare the results on the training set. Since there are two hyper-parameters in the optimization formulation, i.e., C and s, to study the effectiveness of RACOS under different hyper-parameters, we test $s \in \{-1, 0\}$ and $C \in \{0.1, 0.5, 1, 2, 5, 10\}$. We set the total number of calls to the objective function to be $40n$ for all algorithms except for CCCP, while CCCP runs until it converges. Each algorithm is repeated 30 times independently. The achieved objective values are reported in Figure 2.

As shown in Figure 2, compared with SOO, REMBO, and CMA-ES, RACOS has the best performance in all situations. Notice that the smaller the C is, the closer the objective function is to convexity, therefore, the optimization difficulty increases with C. On *USPS+N*, we can observe that CCCP has the best performance when the objective function is very close to convexity (C is very small), since it is a gradient-based method. However, CCCP does not fit well to high non-convexity. It can be further observed that the advantage of RACOS increases as C increases in all situations. This implies that RACOS is suitable for complex tasks.

Conclusion

This paper performs a general theoretical investigation of classification-based optimization methods, where an upper bound of the query complexity of such methods is derived. The upper bound reveals two critical factors of the classification model effecting the optimization performance. Furthermore, we prove that optimization problems with *Local Lipschitz* continuity can be solved in polynomial time by properly configured classification-based optimization methods.

By following the identified critical factors, we propose the *randomized coordinate shrinking* algorithm for learning the classification model, forming the RACOS optimization algorithm for both continuous and discrete domains. The experiments on optimization testing functions and machine learning tasks including spectral clustering and classification with Ramp loss demonstrate that RACOS is more effective than the compared optimization methods. We also notice that RACOS scales better in high-dimensional optimization problems, and has a clearer advantage as the difficulty of the optimization problem increases.

We will test RACOS in more machine learning tasks, and improve RACOS to high-dimensional optimization by, e.g., adapting the random embedding technique (Qian and Yu 2016).

References

Blake, C. L.; Keogh, E.; and Merz, C. J. 1998. UCI Repository of machine learning databases. [http://www.ics.uci.edu/~mlearn/MLRepository.html].

Brochu, E.; Cora, V. M.; and De Freitas, N. 2010. A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv preprint arXiv:1012.2599*.

Bull, A. D. 2011. Convergence rates of efficient global optimization algorithms. *Journal of Machine Learning Research* 12:2879–2904.

Collobert, R.; Sinz, F.; Weston, J.; and Bottou, L. 2006. Trading convexity for scalability. In *Proceedings of the 23rd International Conference on Machine learning (ICML'06)*, 201–208.

de Boer, P.; Kroese, D. P.; Mannor, S.; and Rubinstein, R. Y. 2005. A tutorial on the cross-entropy method. *Annals of Operations Research* 134(1):19–67.

de Freitas, N.; Smola, A. J.; and Zoghi, M. 2012. Exponential regret bounds for gaussian process bandits with deterministic observations. In *Proceedings of the 29th International Conference on Machine Learning (ICML'12)*.

Golberg, D. E. 1989. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading, Massachusetts: Addison-Wesley.

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

Kearns, M. J., and Vazirani, U. V. 1994. *An Introduction to Computational Learning Theory*. Cambridge, Massachusetts: MIT Press.

Larrañaga, P., and Lozano, J. 2002. *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*. Boston, Massachusetts: Kluwer.

Lozano, J. A.; naga, P. L.; Inza, I.; and Bengoetxea, E. 2006. *Towards a New Evolutionary Computation: Advances on Estimation of Distribution Algorithms*. Berlin, Germany: Springer-Verlag.

Mitchell, T. 1997. *Machine Learning*. McGraw Hill.

Mühlenbein, H. 1997. The equation for response to selection and its use for prediction. *Evolutionary Computation* 5(3):303–346.

Munos, R. 2011. Optimistic optimization of a deterministic function without the knowledge of its smoothness. In *Advances in Neural Information Processing Systems 24 (NIPS'11)*, 783–791.

Munos, R. 2014. From bandits to Monte-Carlo Tree Search: The optimistic principle applied to optimization and planning. *Foundations and Trends in Machine Learning* 7(1):1–130.

Neumann, F., and Wegener, I. 2007. Randomized local search, evolutionary algorithms, and the minimum spanning tree problem. *Theoretical Computer Science* 378(1):32–40.

Qian, H., and Yu, Y. 2016. Scaling simultaneous optimistic optimization for high-dimensional non-convex functions with low effective dimensions. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI'16)*.

Qian, C.; Yu, Y.; and Zhou, Z.-H. 2015a. Pareto ensemble pruning. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI'15)*, 2935–2941.

Qian, C.; Yu, Y.; and Zhou, Z.-H. 2015b. Subset selection by pareto optimization. In *Advances in Neural Information Processing Systems 28 (NIPS'15)*.

Vapnik, V. 2000. *The Nature of Statistical Learning Theory*. Springer Science & Business Media.

Von Luxburg, U. 2007. A tutorial on spectral clustering. *Statistics and Computing* 17(4):395–416.

Wang, Z.; Zoghi, M.; Hutter, F.; Matheson, D.; and De Freitas, N. 2013. Bayesian optimization in high dimensions via random embeddings. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI'13)*, 1778–1784.

Yu, Y., and Qian, H. 2014. The sampling-and-learning framework: A statistical view of evolutionary algorithm. In *Proceedings of the 2014 IEEE Congress on Evolutionary Computation (CEC'14)*, 149–158.

Yu, Y.; Yao, X.; and Zhou, Z.-H. 2012. On the approximation ability of evolutionary optimization with application to minimum set cover. *Artificial Intelligence* 180-181:20–33.

Yuille, A. L., and Rangarajan, A. 2001. The concave-convex procedure (CCCP). In *Advances in Neural Information Processing Systems 14 (NIPS'01)*, 1033–1040.