Abstract—Evolutionary algorithms (EAs), a large class of general purpose optimization algorithms inspired from the natural phenomena, are widely used in various industrial optimizations and often show excellent performance. This paper presents an attempt towards revealing their general power from a statistical view of EAs. By summarizing a large range of EAs into the sampling-and-learning framework, we show that the framework directly admits a general analysis on the probable-absolute-approximate (PAA) query complexity. We particularly focus on the framework with the learning subroutine being restricted as a binary classification, which results in the sampling-and-classification (SAC) algorithms. With the help of the learning theory, we obtain a general upper bound on the PAA query complexity of SAC algorithms. We further compare SAC algorithms with the uniform search in different situations. Under the error-target independence condition, we show that SAC algorithms can achieve polynomial speedup to the uniform search, but not super-polynomially; while the one-side-error condition further allows a super-polynomial speedup. Under the one-side-error condition, we show that super-polynomial speedup can be achieved. This work only touches the surface of the framework. Its power under other conditions is still open.

I. INTRODUCTION

In many practical optimization problems, the objective functions are hidden or too complicated to be analyzed. Under this kind of circumstances, direct optimization algorithms are appealing, which follows the trial-and-error style with some heuristics. Evolutionary algorithms (EAs) [3] are a large family of such algorithms. The family includes genetic algorithms [17], evolutionary programming [26], evolutionary strategies [5], and also covers other nature-inspired heuristics including particle swarm optimization [25], ant colony optimization [11], estimation of distribution algorithms [29], etc.

Theoretical studies of EAs have been developed rapidly in the recent decades, particularly noticeable of the blooming of running time analysis [32], [2], [20]. With the development of several analysis techniques (e.g. [18], [44], [9], [37]), EAs have been theoretically investigated on problems from simple synthetic ones (e.g. [13]) to combinatorial problems (e.g. [35]) as well as NP-hard problems (e.g. [43]). During these analyses, effects of EAs components have been disclosed [42], including the crossover operators (e.g. [22], [31], [8], [33]), the population size (e.g. [21], [20], [40], [6]), etc. Measures of the performance also have developed to cover the approximation complexity (e.g. [19], [16], [43], [25]), the fixed-parameter complexity (e.g. [27], [33]), the complexity under fixed-budget computation [23], etc. While most of these analyses studied instances of EAs on problem cases, general performance analysis may even be more desired, as the application of EAs is nearly unlimited. The famous No-Free-Lunch Theorem [41] used a quite general framework of EAs and gave a general conclusion that any two EAs are with the same performance (at least on discrete domains) given no prior knowledge of the problem distribution, of which the general running time is exponential [44]. When the complexity of a problem class is bounded, a general convergence lower bound can be derived for a class of EAs [15]. For more general EAs, the Black-Box model can derive the best possible performance [12], [1], [30]. We have learned that a general performance analysis relies on a general framework of EAs.

It has been noticed that various implementations of EAs share a common structure that consists of a cycle of sampling and model building [47]. In this work, we propose to study the sampling-and-learning (SAL) framework. EAs commonly employ some heuristic to reproduce solutions, which is captured by the sampling step of SAL; and they also distinguish the quality of the reproduced solutions to guide the next sampling (e.g., genetic algorithms remove a portion of the worst solutions), which is captured by the learning step of SAL. The SAL framework can simulate a wide range of EAs as well as other heuristic search methods, by specifying the sampling and the learning strategies.

We evaluate this framework by the probable-absolute-approximate (PAA) query complexity. PAA complexity counts the number of fitness evaluations before reaching to an approximate solution with a probability, which is close to the intuitive evaluation of EAs in practice. We show that the SAL framework immediately admits a general PAA upper bound. For a specific version of SAL that uses classification algorithms, named the SAC algorithms, we obtain a tighter PAA upper bound by incorporating the learning theory results. Further comparing with the uniformly random search, we disclose that, under the error-target independence condition, SAC algorithms can polynomially reduce the complexity of the uniform search, but not super-polynomially; while the one-side-error condition further allows a super-polynomial improvement. This study shows that the classification error is an important effecting factor, which was not noticed before. We also notice that a good learning algorithm may not be necessary for a good SAL algorithm.

The rest of this paper is organized as follows: Section II introduces the SAL framework. In Section III, we compare the SAC algorithms, a specific version of the SAL framework, with the uniform search. Finally, Section IV concludes the paper.
II. THE SAMPLING-AND-LEARNING FRAMEWORK

In this paper, we consider general minimization problems $f$. We always denote $X$ as the whole solution space which an algorithm will search among. In the analysis of this paper, we consider $X \subseteq \mathbb{R}^n$ is a compact set (in the Euclidean space, the compact set is equivalent to the bounded and closed set) and $f : X \rightarrow \mathbb{R}$ is a continuous function. Thus there must exist at least one solution $x^* \in X$ such that $f(x^*) = \min_{x \in X} f(x)$. We use $D$ to denote sub-regions of $X$ and define $|D| = \int_D 1 \, dx$. For the sake of the convenience for the analysis, we assume without loss of generality that $|X| = 1$ since $X$ is a bounded and closed set. Denote $D_\alpha = \{x \in X | f(x) \leq \alpha\}$ for any scalar $\alpha, UX$ as the uniform distribution over $X, T$ and $D$ as the probability distributions. Besides, by $\text{poly}(\cdot)$, we mean the set of all polynomials with the related variables, and by $\text{superpoly}(\cdot)$, we mean the set of all functions that grow faster than any function in $\text{poly}(\cdot)$ with the related variables.

**Definition 1 (Minimization Problem)** A minimization problem consists of a continuous solution space $X$ and a continuous function $f : X \rightarrow \mathbb{R}$, where $X \subseteq \mathbb{R}^n$ and $X$ is a compact set. The goal is to find a solution $x^* \in X$ such that $f(x^*) \leq f(x)$ for all $x \in X$.

Since $X$ is a compact set and $f$ is a continuous function, there must exist one solution $x^* \in X$ such that $f(x^*) = \max_{x \in X} f(x)$. Namely, $f$ is bounded in $[f(x^*), f(x')]$. Therefore, in the rest of the paper, we assume without loss of generality that the value of $f$ is bounded in $[0, 1]$, i.e., $\forall x \in X : f(x) \in [0, 1]$. Given an arbitrary function $g$ with bounded value range over the input domain, the bound can be implemented by a simple normalization $f(x) = \frac{g(x) - g(x^*)}{\max_{x \in X} g(x) - g(x^*)}$. Thus we assume in the rest of this paper that every minimization problem has its minimum value 0.

In real-world applications, we expect EAs to achieve some good enough solutions with a not quite small probability, which corresponds to approximation (e.g. [43]) and probabilistic performance (e.g. [43]). Combining the two, we study the probable-absolute-approximate (PAA) query complexity, which is the number of fitness evaluations that an algorithm takes before reaching an approximate quality, as defined in Definition 2. The PAA query complexity closely reflects our intuitive evaluation of EAs in practice.

**Definition 2 (Probable-Absolute-Approximate Query Complexity)** Given a minimization problem $f$, an algorithm $A$, and any $0 < \delta < 1$ as well as any approximation level $\alpha^* > 0$, then the probable-absolute-approximate (PAA) query complexity is the number of calls to $f(\cdot)$ such that, with probability at least $1 - \delta$, $A$ finds a solution $x$ with $f(x) \leq \alpha^*$.

**A. The General Framework**

Most EAs share a common trial-and-error structure with several important properties:

- a) directly access the solution space, generate solutions, and evaluate the solutions;
- b) the generation of new solutions depends only on a short history of past solutions;
- c) both “global” and “local” heuristic operators are employed to generate new solutions.

We present a sampling-and-learning (SAL) framework in Algorithm 1 to capture these properties. The SAL framework starts from a random sampling in Step 1 like all EAs. Steps 2 and 13 record the best-so-far solutions throughout the search. SAL follows a cycle of learning and sampling stages. In Step 7, it learns a hypothesis $h_t$ (i.e., a mapping from $X$ to $\mathbb{R}$) via the learning algorithm $\mathcal{L}$. Note that the learning algorithm allows to take the current data set $T_t$, the last data set $T_{t-1}$, and the last hypothesis $h_{t-1}$ into account. Different EAs may make different use of them. Step 8 initializes the sample set for the next iteration. The sample set can be initialized as an empty set, or to preserve some good solutions from the previous iteration. In Steps 9 to 12, it samples from the distribution transformed from the hypothesis as well as from the whole solution space balanced by a probability. The distribution $T_h$ implies the potential good regions learned by $h_t$.

It should be noted that the SAL framework is not a concrete optimization algorithm but an abstract summary of a range of EAs, nor does the learning stage of the framework imply an accurate learning. We explain in the following how we could mimic several different EAs by the SAL framework. It is noticeable that the explanation is not a rigorous proof, but an intuitive illustration that the SAL framework can correspond to various implementations.

**Algorithm 1** The sampling-and-learning (SAL) framework

**Input:**
- $\alpha^* > 0$: Approximation level
- $T \in \mathbb{N}^+$: Number of iterations
- $m_0, \ldots, m_T \in \mathbb{N}$: Number of samples
- $\lambda \in [0, 1]$: Balancing parameters
- $\mathcal{L}$: Learning algorithm
- $T$: Distribution transformation of hypothesis

**Procedure:**
1. Collect $S_0 = \{x_1, \ldots, x_{m_0}\}$ by i.i.d. sampling from the uniform distribution over $X$
2. $\tilde{x} = \arg\min_{x \in S_0} f(x)$
3. Initialize the hypothesis $h_0$
4. $T_0 = \emptyset$
5. for $t = 1$ to $T$
6. Construct $T_t = \{(x_1, y_1), \ldots, (x_{m_{t-1}}, y_{m_{t-1}})\}$, where $x_i \in S_{t-1}$ and $y_i = f(x_i)$
7. $h_t = \mathcal{L}(T_t, T_{t-1}, h_{t-1}, t)$, the learning step
8. Initialize $S_t$ from $T_t$
9. for $i = 1$ to $m_t$
10. Sample $x_i$ from $\{T_{h_t}, \mathcal{U}_X\}$ with probability $\lambda$
11. $S_t = S_t \cup \{x_i\}$
12. end for
13. $\tilde{x} = \arg\min_{x \in S_t \cup \{\tilde{x}\}} f(x)$
14. end for
15. return $\tilde{x}$
where $\Pr_u = \int_{D_\alpha} \mathcal{U}(x) \, dx$ is the success probability of uniform sampling.

$$\Pr_h = \frac{\sum_{t=1}^{T} m_t \cdot \Pr_{h_t}}{\sum_{t=1}^{T} m_t} = \frac{\sum_{t=1}^{T} m_t \cdot \int_{D_\alpha} \mathcal{T}_h(x) \, dx}{\sum_{t=1}^{T} m_t}$$

is the average success probability of sampling from the learnt hypothesis, $m_{Pr_{h_t}}$ is the required sample size realizing $\Pr_{h_t}$, and $D_\alpha$ is the set $\{x \in X | f(x) \leq \alpha^*\}$.

**Proof.** Let $m_0$ be the initial sample size. In every iteration, we need $m_{Pr_{h_t}}$ samples to realize the probability $\Pr_{h_t}$ (generally the higher the probability the larger the sample size, but it depends on the concrete implementation of the algorithm), thus $\sum_{t=1}^{T} m_{Pr_{h_t}}$ number of samples is naturally required. We prove the rest of the bound.

Let’s consider the probability that after $T$ iterations, the SAL algorithm outputs a bad solution $x$ such that $f(x) > \alpha^*$. Since the $x$ is the best solution among all sampled examples, the probability is the intersection of events that every step of the sampling does not generate such a good solution.

1. For the sampling from uniform distribution over the whole solution space $X$, the probability of failure is $1 - \Pr_u$.
2. For the sampling from the learnt hypothesis $h_t$, according to the distribution $\mathcal{T}_h$, the probability of failure is denoted as $1 - \Pr_{h_t}$.

Since every sampling is independent, we can expand the probability of overall failures, i.e., for any solution $x$ belongs to the all sampled examples,

$$\Pr(f(x) > \alpha^*) = (1 - \Pr_u)^m_0.$$ 

$$\prod_{t=1}^{T} \left(1 - \Pr_{h_t}\right)^m_i = \prod_{t=1}^{T} \left(1 - \left(1 - \Pr_{h_t}\right)^m_i \right) \leq e^{-\Pr_{h_t}} \int_{D_\alpha} \mathcal{T}_h(x) \, dx.$$ 

Let’s consider the probability of failure after $T$ iterations. The algorithm needs to use the initial hypothesis resulting the same sampling distribution as that from the initial iteration. Let $S_t$ be the ordered set to contain the globally best particle and the personally best particles in Step 8. The learning algorithm in the SAL algorithm can be set to utilize the current data set and the last data set to recover the velocity, and utilize the last hypothesis and the globally and personally best particles recorded through $S_t$ to generate the new hypothesis that simulates the movement of particles in the PSO.

Overall, the SAL framework captures the trial-and-error structure as well as the global–local search balance, while leaving the details of the local sampling distribution being implemented by different heuristics.

The SAL framework directly admits a general upper bound of the PAA query complexity, as stated in Theorem 1

**Theorem 1** For any minimization problem $f$ and any approximation level $\alpha^* > 0$, with probability at least $1 - \delta$, a SAL algorithm will output a solution $x$ with $f(x) \leq \alpha^*$ using $m_{SE}$ number of queried samples bounded from above by

$$O\left(m_0 + \max\left\{ \frac{1}{(1 - \lambda)\Pr_u + \lambda\Pr_{h_t}}, \ln\frac{1}{\delta}, \frac{1}{\delta m_{Pr_{h_t}}} \right\} \right).$$
data set $T$ in the learning algorithm, but not the last data set $T'$ and the last hypothesis $h'$. Putting Algorithm 2 into the framework of Algorithm 1, we always set $S_t = \emptyset$ for SAC, and $T_h$ will be some distribution over the positive area of $h$.

By these specifications, we can have a general PAA performance for SAC algorithms. According to Theorem 1, we need to estimate a lower bound of $\mathbb{P}[h]$, i.e., how likely the distribution $T_h$ will lead to a good solution. Recall $D_\alpha = \{ x \in X | f(x) \leq \alpha \}$ for any scalar $0 < \alpha < 1$. Denote $D_h = \{ x \in X | h(x) = +1 \}$ for any hypothesis $h$, $\mathcal{U}_D(h)$ as the uniform distribution over $D_h$, and $D_{KL}$ as the Kullback-Leibler (KL) divergence. KL-divergence measures how different one distribution departs from another one. For probability distributions $P$ and $Q$ of two continuous random variables, $D_{KL}(P||Q) = \int_{-\infty}^{+\infty} \ln \left( \frac{p(x)}{q(x)} \right) p(x)dx$, where $p(x)$ and $q(x)$ are the probability densities of $P$ and $Q$. Let $D$ denote the symmetric difference operator of two sets. We have a lower bound of the success probability as in Lemma 1.

**Lemma 1** For any minimization problem $f$, any approximation level $\alpha > 0$, any hypothesis $h$, the probability that a solution sampled from an arbitrary distribution $T_h$ defined on $D_h$ will lead to a solution in $D_{\alpha}$ is lower bounded as

$$\mathbb{P}_h \geq \frac{|D_{\alpha} \cap D_h|}{|D_h|} - \frac{1}{2} D_{KL}(T_h||\mathcal{U}_D)$$

*Proof.* Let $I[\cdot]$ denote the indicator function, namely, $I[\text{true}] = 1$ and $I[\text{false}] = 0$. The proof starts from the definition of the probability,

$$\mathbb{P}_h = \int_{D_h} T_h(x) \cdot I[x \in D_{\alpha}] dx = \int_{D_h} (T_h(x) - \mathcal{U}_D(x)) \cdot I[x \in D_{\alpha}] dx = \frac{|D_{\alpha} \cap D_h|}{|D_h|} + \int_{D_h} (T_h(x) - \mathcal{U}_D(x)) \cdot I[x \in D_{\alpha}] dx \geq \frac{|D_{\alpha} \cap D_h|}{|D_h|} + \sup_{x \in D_h} (T_h(x') - \mathcal{U}_D(x')) \cdot I[x \in D_{\alpha}] dx \geq \frac{|D_{\alpha} \cap D_h|}{|D_h|} - \frac{1}{2} D_{KL}(T_h||\mathcal{U}_D) = \frac{|D_{\alpha} \cap D_h|}{|D_h|} - \frac{1}{2} D_{KL}(T_h||\mathcal{U}_D),$$

where the last inequality is by Pinsker’s inequality.

We cannot pre-determine $D_h$, but we know that $h$ is derived by a binary classification algorithm from a data set which is labeled according to the threshold parameter $\alpha$. For the binary classification, we know that the generalization error, which is the expected misclassification rate, can be bounded above by the training error, which is the misclassification rate in the seen examples, as well as the generalization gap involving the complexity of the hypothesis space $\mathcal{H}$, as in Lemma 2. The $VC(\mathcal{H})$ is the VC-dimension measuring the complexity of $\mathcal{H}$.

**Lemma 2** Let $\mathcal{H} = \{ h : X \rightarrow \{-1,+1\} \}$ be the hypothesis space containing a family of binary classification functions and $VC(\mathcal{H}) = d$, if there exist $m$ samples i.i.d. from $X$ according to some fixed unknown distribution $D$, then, $\forall h \in \mathcal{H}$ and $\forall 0 < \eta < 1$, the following upper bound holds true with probability at least $1 - \eta$:

$$\epsilon_D \leq \hat{\epsilon}_D + \sqrt{8m^{-1}(d \log (2emn^{-d}) + \log (4\eta^{-1}))}$$

where $\epsilon_D$ is the expected error rate of $h$ over $D$ and $\hat{\epsilon}_D$ is the error rate in the sampled examples from $D$, and when $\hat{\epsilon}_D = 0$, $\epsilon_D \leq 2m^{-1}(d \log (2emn^{-d}) + \log (2\eta^{-1}))$.

Again by Pinsker’s inequality, we know that the error $\epsilon_D$ under the distribution $D$ can be converted to the error $\epsilon_U$ under the uniform distribution, as

$$\epsilon_U \leq \frac{\epsilon_D}{1 - \sqrt{\frac{1}{2} D_{KL}(D||\mathcal{U})}} \leq \frac{\epsilon_D + \sqrt{8m^{-1}(d \log (2emn^{-d}) + \log (4\eta^{-1}))}}{1 - \sqrt{\frac{1}{2} D_{KL}(D||\mathcal{U})}},$$

where we only take the event that the generalization inequality holds with probability $1 - \eta$ into account. For simplicity, we denote the right-hand part as $\Psi_{m,\eta}^{\epsilon_D,d,D_{KL}(D||\mathcal{U})}$, which decreases with $m$ and $\eta$, and increases with $\epsilon_D$, $d$, and $D_{KL}(D||\mathcal{U})$.

We can use this inequality to eliminate the $D_h$ in Lemma 1. In every iteration of SAC algorithms, there are $m_t$ samples collected, which make the error of $h_t$ bounded.

**Theorem 2** For any minimization problem $f$, any constant $0 < \eta < 1$, and any approximation level $\alpha > 0$, the average success probability of sampling from the learned hypothesis of any SAC algorithm is lower bounded as

$$\mathbb{P}_h \geq \sum_{t=1}^{T} m_t \sum_{m=1}^{m_t} \left( \frac{|D_{\alpha}| - 2\Psi_{m,\eta}^{\epsilon_D,d,D_{KL}(D||\mathcal{U})}}{|D_{\alpha}|} \right) - \frac{1}{2} D_{KL}(T_h||\mathcal{U}_D),$$

where $D_t = \lambda T_{h_t} + (1 - \lambda) \mathcal{U}_X$ is the sampling distribution at iteration $t$, $\hat{\epsilon}_D$ is the training error rate of $h_t$, $d$ is the VC-dimension of the learning algorithm.

*Proof.* By set operators,

$$|D_{\alpha} \cap D_{h_t}| = |D_{\alpha} \cup D_{h_t}| - |D_{\alpha} \Delta D_{h_t}| \geq |D_{\alpha} \cup D_{h_t}| - |D_{\alpha} \Delta D_{h_t}| - |D_{\alpha} \Delta D_{h_t}| = |D_{\alpha} \cup D_{h_t}| - |D_{\alpha} \Delta D_{h_t}| - \epsilon_{\mathcal{U},t}.$$
the average success probability, the SAC algorithm improves from the uniform search depends on \(\lambda\) algorithms relative to the uniform search. In other words, the uniform search is the SAL algorithm with space always by sampling solutions uniformly at random. In could design various distributions for relative variables that generally effect the complexity. One though the expression looks sophisticated, it can still reveal that the SAC algorithm is worse. We are then interested in investigating the conditions under which SAC algorithms can accelerate from the uniform search.

A. A Polynomial Acceleration Condition

**Condition 1** (Error-Target Independence) In SAC algorithms, for any \(t\) and any approximation level \(\alpha^* > 0\), when sampling a solution \(x\) from \(U_X\), the event \(x \in D_{hi} \Delta D_{ai}\) and the event \(x \in D_{ai}\) are independent.

We call SAC algorithms that are under the error-target independence condition as SAC\(_i\) algorithms. The condition is defined using the independence of random variables. From the set perspective, it is equivalent with

\[
|D_{ai} \cap (D_{ai} \Delta D_{hi})| = |D_{ai}| \cdot |(D_{ai} \Delta D_{hi})|
\]

Under the condition, we can bound from below the probability of sampling a good solution, as stated in Lemma 3

**Lemma 3** For SAC\(_i\) algorithms, it holds for all \(t\) that

\[
\frac{|D_{ai} \cap D_{hi}|}{|D_{hi}|} \geq \frac{|D_{ai}|(1 - \epsilon_{\delta \cdot t})}{|D_{ai}| + \epsilon_{\delta \cdot t}},
\]

where \(\epsilon_{\delta \cdot t}\) is the expected error rate of \(h_t\) under \(U_X\).

**Proof.** For the numerator,

\[
|D_{ai} \cap D_{hi}| = |D_{ai}| - |D_{ai} \cap (D_{ai} \Delta D_{hi})|
\]

\[
= |D_{ai}| - |D_{ai}| \cdot |D_{ai} \Delta D_{hi}|
\]

\[
\geq |D_{ai}|(1 - \epsilon_{\delta \cdot t}),
\]

where the first equation is by \(D_{ai} \subseteq D_{ai}\), and the second equality is by the error-target independence condition.

For the denominator, we consider the worst case that all errors are out of \(D_{hi}\) and thus \(|D_{hi}| \leq |D_{ai}| + \epsilon_{\delta \cdot t}\).

Similar to Theorem 2 we can bound from below the average success probability of sampling from the positive area of the learnt hypothesis,

\[
\Pr_h \geq \frac{1 - \eta}{\sum_{t=1}^{T} m_t} \sum_{t=1}^{T} m_t \left( \frac{|D_{ai}|(1 - \epsilon_{\delta \cdot t})}{|D_{ai}| + \epsilon_{\delta \cdot t}} - |D_{ai}| \right) \frac{1}{2} D_KL(T_h || U_{D_{hi}})\right).
\]

We compare the uniform search with the SAC\(_i\) algorithms using uniform sampling within \(D_{hi}\), i.e., \(D_KL(T_h || U_{D_{hi}}) = 0\), which is an optimistic situation. Then by Lemma 3

\[
\Pr_h \geq \frac{1 - \eta}{\sum_{t=1}^{T} m_t} \sum_{t=1}^{T} m_t \left( \frac{|D_{ai}|(1 - \epsilon_{\delta \cdot t})}{|D_{ai}| + \epsilon_{\delta \cdot t}} \right).
\]

By plugging \(\epsilon_{\delta \cdot t} \leq \frac{\epsilon_{D_t}}{1 - \sqrt{\frac{0}{2} D_KL(D_t || U_X)}} = Q \cdot \epsilon_{D_t}\), where \(\epsilon_{D_t}\) is the expected error rate of \(h_t\) under the distribution \(D_t = \lambda U_{D_{hi}} + (1 - \lambda) U_X\) and \(Q = (1 - \sqrt{\frac{0}{2} D_KL(D_t || U_X)})^{-1}\),

\[
\Pr_h \geq \frac{1 - \eta}{\sum_{t=1}^{T} m_t} \sum_{t=1}^{T} m_t \left( \frac{|D_{ai}|(1 - \epsilon_{Q \cdot \epsilon_{D_t}})}{|D_{ai}| + \epsilon_{Q \cdot \epsilon_{D_t}}} \right).
\]

Note from Lemma 2 that, the convergence rate of the error is \(O\left(\frac{1}{m}\right)\) ignoring other variables and logarithmic terms from Lemma 2 We assume that SAC\(_i\) uses learning algorithms.
with convergence rate \( \tilde{O}(\frac{1}{m^2}) \). We then find that such SAC\(_1\) algorithms cannot exponentially improve the uniform search in the worst case, as Proposition 1.

**Proposition 1** Using learning algorithms with convergence rate \( \Theta(\frac{1}{n}) \), \( \forall f, \alpha^* > 0 \) and \( 0 < \delta < 1 \), with probability at least \( 1 - \delta \), if the query complexity of the uniform search is superpoly\(\left(\frac{1}{\alpha^*}, n, \frac{1}{\delta}\right)\), the query complexity of SAC\(_1\) algorithms is also superpoly\(\left(\frac{1}{\alpha^*}, n, \frac{1}{\delta}\right)\) in the worst case.

**Proof.** The query complexity of the uniform search being superpoly\(\left(\frac{1}{\alpha^*}, n, \frac{1}{\delta}\right)\) implies that

\[
\frac{1}{\Pr_{\alpha}} = \frac{1}{|D_{\alpha^*}|} = \text{superpoly}\left(\frac{1}{\alpha^*}, n, \frac{1}{\delta}\right).
\]

For the SAC\(_1\) algorithms, if we ask the learning algorithm to produce a classifier with error rate \( \text{superpoly}\left(\frac{1}{\alpha^*}, n, \frac{1}{\delta}\right) \), it will require superpoly\(\left(\frac{1}{\alpha^*}, n, \frac{1}{\delta}\right)\) number of samples in the worst case, so that the proposition holds. To avoid this, we can only expect the error rate to be \( \text{poly}\left(\frac{1}{\alpha^*}, n, \frac{1}{\delta}\right) \) in order to keep the query complexity at each iteration small.

Meanwhile, we can only have \( T = \text{poly}\left(\frac{1}{\alpha^*}, n, \frac{1}{\delta}\right) \) iterations otherwise we will have super-polynomial number of samples.

Following the optimistic case of Eq.(1), since \( Q \geq 1 \), we consider one more optimistic situation that \( Q = 1 \). Let \( \eta = 0.5 \). Even though, in the worst case that \( |D_{h_{\eta}}| = |D_{\alpha^*}| + Q\epsilon_{D_{\eta}} \), we can have that

\[
\Pr_{h_{\eta}} = \frac{1}{2} \sum_{t=1}^{T} m_{t} \left( \frac{|D_{\alpha^*}| (1 - \epsilon_{D_{\eta}})}{|D_{\alpha^*}| + \epsilon_{D_{\eta}}} \right) = \frac{1}{\text{poly}\left(\frac{1}{\alpha^*}, n, \frac{1}{\delta}\right)} \frac{1}{\text{superpoly}\left(\frac{1}{\alpha^*}, n, \frac{1}{\delta}\right)},
\]

where it is noted that as long as \( \epsilon_{D_{\eta}} = \text{poly}\left(\frac{1}{\alpha^*}, n, \frac{1}{\delta}\right) \) the value of \( |D_{\alpha^*}| \) cannot affect the result. Then substituting \( \Pr_{h_{\eta}} \) into Theorem 1 obtains the total samples \( m_{\epsilon} = \text{superpoly}\left(\frac{1}{\alpha^*}, n, \frac{1}{\delta}\right) \) that proves the proposition.

The proposition implies that the SAC\(_1\) algorithms can face the same barrier as that of the uniform search. Nevertheless, the SAC\(_1\) algorithms can still improve the uniform search within a polynomial factor. We show this by case studies.

**On Sphere Function Class:**

Given the solution space \( X_n = \{(x_1, \ldots, x_n) \mid \forall i = 1, \ldots, n : x_i \in [0, 1]\} \), the Sphere Function class is \( F_{\text{sphere}}^n = \{ f_{\text{sphere}}^n \mid \forall x^* \in X_n \} \) where

\[
f_{\text{sphere}}^n(x) = \frac{1}{n} \| x - x^* \|_2^2 = \frac{1}{n} \sum_{i=1}^{n} (x_i - x_i^*)^2.
\]

Obviously, \( |X_n| = 1 \), \( f_{\text{sphere}}^n \in [0, 1] \) is convex, and the optimal value is 0. It is important to note that the volume of a \( n \)-dimensional hyper-sphere with radius \( r \) is \( \frac{\pi^{n/2}}{\Gamma(\frac{n}{2} + 1)} r^n \), where \( \Gamma(s) = \int_0^\infty t^{s-1} e^{-t} dt \), so that \( |D_{\alpha^*}| = \left(\frac{\pi^{n/2}}{\Gamma(\frac{n}{2} + 1)} (n\alpha)^{n/2} = C_n(\alpha)^{n/2} \right) \) for any \( \alpha > 0 \), where \( C_n = \Theta \left( \frac{(2\pi e)^{\frac{n}{2}}}{\sqrt{n}} \right) \), since the radius leading to \( f_{\text{sphere}}^n(x) = \frac{1}{n} \| x - x^* \|_2^2 \leq \alpha \) is \( \sqrt{n\alpha} \).

Note that \( \Pr_{\alpha} = |D_{\alpha^*}| = C_n(\alpha^*)^{n/2} \), with approximation level \( \alpha^*> 0 \). It is straightforward to obtain that, minimizing any function in \( F_{\text{sphere}}^n \) using the uniform search, the PAA query complexity with approximation level \( \alpha^*> 0 \) is, with probability at least \( 1 - \delta \),

\[
O \left( \frac{1}{\alpha^*} \frac{n}{\delta} \ln \frac{1}{\delta} \right).
\]

We assume \( L_{\text{sphere}} \) is a learning algorithm that searches in the hypothesis space \( H_n \) consisting of all the hyper-spheres in \( \mathbb{R}^n \) to find a sphere that is consistent with the training data, and meanwhile the sphere satisfies the error-target independence condition. Then a SAC algorithm using \( L_{\text{sphere}} \) is a SAC\(_1\) algorithm. We simply assume that the search of the consistent sphere is feasible. Note that \( VC(H_n) = n + 1 \).

**Lemma 4** For any \( h_{\eta} \), denote \( \epsilon_{\text{LD}} \) as the error rate of \( h_{\eta} \) under the uniform distribution over \( X \) and \( \epsilon_{D_{\eta}} \) as the error rate of \( h_{\eta} \) under the distribution \( D_{\eta} = \mathcal{U}D_{h_{\eta}} + (1 - \lambda)D_X \), then it holds that

\[
\epsilon_{\text{LD}} \leq \frac{1}{1 - \lambda^*} \epsilon_{D_{\eta}},
\]

where \( \lambda \in [0, 1] \) and \( D_{h_{\eta}} \) is the uniform distribution over \( D_{h_{\eta}} \).

**Proof.** Let \( I[\cdot] \) be the indicator function and \( D_{\#} \) be the area where \( h_{\eta} \) makes mistakes. We split \( D_{\#} \) into \( D_{\#}^+ = D_{\#} \cap D_{h_{\eta}} \) and \( D_{\#}^- = D_{\#} \setminus D_{h_{\eta}}^+ \). We can calculate the probability density that \( D_{\#}^+(x) = \lambda \frac{|D_{h_{\eta}}|}{|X|} + (1 - \lambda) \frac{|D_{\eta}|}{|X|} \frac{1}{|X|} \) for any \( x \in D_{\#}^+ \), and \( D_{\#}^-(x) = (1 - \lambda) \frac{|X\backslash D_{h_{\eta}}|}{|X|} \frac{1}{|X|} \) for any \( x \in D_{\#}^- \).

Thus,

\[
\epsilon_{D_{\eta}} = \int_{X} D_{\#}(x) I[h_{\eta} \text{ makes mistake on } x] dx
= \int_{D_{\#}^+} D_{\#}(x) dx + \int_{D_{\#}^-} D_{\#}(x) dx
\geq \int_{D_{\#}^+} (1 - \lambda) \frac{1}{|X|} dx + \int_{D_{\#}^-} (1 - \lambda) \frac{1}{|X|} dx
= (1 - \lambda) \epsilon_{\text{LD}},
\]

which proves the lemma.

We then obtain the PAA complexity as in Proposition 2.

**Proposition 2** For any function in \( F_{\text{sphere}}^n \) and any approximation level \( \alpha^*> 0 \), SAC\(_1\) algorithms can achieve the PAA query complexity, for any \( n \geq 2 \),

\[
O \left( \frac{1}{\alpha^*} \left( \frac{n}{\delta} \log \frac{1}{\sqrt{\alpha}} \right) \log \left( \frac{1}{\delta} + n \log \frac{1}{\sqrt{\alpha}} \right) \right)
\]

with probability at least \( 1 - \delta \).

**Proof.** We choose \( \alpha_t = \frac{1}{\sqrt{t}} \) for all \( t \), and use the number of iterations \( T \) to approach \( |D_{\alpha^*}| = \sqrt{|D_{\alpha^*}|} \), for the approximation level \( \alpha^* \). Solving this equation with the sphere volume results in \( T = \log \left( \frac{C_n(\alpha^*^{1/4})}{\sqrt{\alpha^*}} \right) \). We let the SAC\(_1\) algorithm run \( T = \log \frac{1}{\sqrt{\alpha^*}} \) number of iterations. We assume \( \log \frac{1}{\sqrt{\alpha^*}} \) is an integer for simplicity, which does not affect the generality.
In iteration \( t \), using \( L_{\text{sphere}} \), we want the error of the hypothesis \( h_t, \epsilon_{D_t} \), to be \( \frac{1}{2^t} \). Since the \( L_{\text{sphere}} \) produces a hypothesis with zero training error, from 

\[
\epsilon_{D_t} = \frac{1}{2^t} \leq 2m^{-1} \left( d \log (2emd^{-1}) + \log (2n^{-1}) \right),
\]
we can solve the required sample size with \( \eta \) being a constant,

\[
m_t \leq m_T = O(nT2^T) = O \left( \frac{n}{\sqrt{\alpha}} \log \frac{1}{\sqrt{\alpha}} \right)
\]
using the inequality \( \log x \leq cx - \log (c+1) \) for any \( x > 0 \) and any \( c > 0 \). We thus obtain \( \sum_{t=1}^T m_t = O \left( \frac{n}{\sqrt{\alpha}} \log \left( \frac{1}{\sqrt{\alpha}} \right)^2 \right) \).

We then follow Eq. [I]. We use uniform sampling within \( D_{h_t} \), then \( Q = \frac{1}{1-3} \). Letting the SAC1 algorithms use \( m_T \) number of samples in every iteration, \( \lambda = 0.5 \) and \( \eta = 0.5 \), we have

\[
\text{Pr}_k \geq \frac{1}{2 \log \frac{1}{\sqrt{\alpha}}} \sum_{t=1}^{\log \frac{1}{\sqrt{\alpha}}} \left( \frac{\sum_{t=1}^{\log \frac{1}{\sqrt{\alpha}}} (|D_{\alpha^*}|(1 - Q_{D_{\alpha^*}}) + Q_{D_{\alpha^*}})}{C_n \epsilon_{D_t}} \right)
= \frac{C_n \epsilon_{D_t}^2}{2 \log \frac{1}{\sqrt{\alpha}}} \left( \frac{1}{\epsilon_{D_t}} \right)
\]


Fig. 1. The landscape of function \( f_{\text{spike}}^{\alpha, t}(x) \) in \([-1, 1]\).

It is easy to know \( \min_{x \in X_n} f(x) = 0 \) and \( \max_{x \in X_n} f(x) \leq 1 \) for \( f \in F_n \). For any \( \alpha > 0 \), we can bound the area \( |D_{\alpha}| = C_n \alpha^n, C_n(3\alpha)^n \), where \( C_n = \Theta \left( \frac{(2\pi e)^2}{\sqrt{\pi n}} \right) \). The Spike functions are non-convex and non-differentiable with some local optima, as depicted in Figure [I].

Minimizing any function in \( F_n^{\alpha} \) using the uniform search, the PAA query complexity with approximation level \( \alpha > 0 \), is, with probability at least \( 1 - \delta \),

\[
O \left( \frac{1}{\alpha^*} \right)^n \log \frac{1}{\delta}.
\]

We configure the SAC1 algorithm to use the learning algorithm \( L_{\text{spike}} \) that searches the smallest sphere covering all the samples labeled as positive, of which the VC-dimension is \( n + 1 \). Note that since the function is non-convex, the \( L_{\text{spike}} \) may output a sphere that also covers some negative examples, and thus with some training error. Using this SAC1 algorithm to minimize any member in the function class \( F_n^{\alpha} \), we obtain the PAA query complexity as in Proposition [I].

**Proposition 3** For any function in \( F_n^{\alpha} \) and any approximation level \( \alpha > 0 \), SAC1 algorithms can achieve the PAA query complexity

\[
O \left( \frac{1}{\alpha^*} \right)^n \frac{1}{\log \frac{1}{\delta}} \left( \frac{1}{\delta} + n \frac{1}{\sqrt{\alpha^*}} \right),
\]

with probability at least \( 1 - \delta \).

**Proof.** For any function in \( F_n^{\alpha} \), we note that the function is convex in \( D_{\alpha} \) when \( \alpha \) is smaller than 0.05. We set \( \alpha_t = \frac{1}{2^t} \), so that \( t > 5 \), the SAC1 algorithm with \( L_{\text{spike}} \) will deal with a convex function and thus the training error is zero. We use the number of iterations \( T \) to achieve \( |D_{\alpha T}| = \sqrt{|D_{\alpha}|} \). Since \( |D_{\alpha}| \in [C_n \alpha^n, C_n(3\alpha)^n] \), we can obtain \( T \geq \log \frac{2\sqrt[3]{C_n}}{\sqrt{\alpha}} \). We let the SAC1 algorithm run \( T = \log \frac{1}{\sqrt{\alpha}} \) number of iterations and assume that \( \log \frac{1}{\sqrt{\alpha}} \) is an integer.

In iteration \( t \geq 5 \), we want the error of the hypothesis \( h_t, \epsilon_{D_t} \), to be \( \frac{1}{2^t} \). Since the training error can be zero, we can solve the required sample size \( m_t \leq m_T = O \left( \frac{n}{\sqrt{\alpha}} \log \frac{1}{\sqrt{\alpha}} \right) \). We thus obtain \( \sum_{t=1}^T m_t = O \left( \frac{n}{\sqrt{\alpha}} \log \left( \frac{1}{\sqrt{\alpha}} \right)^2 \right) \).

We then follow Eq. [I]. We use uniform sampling within \( D_{h_t} \), then \( Q = \frac{1}{1-3} \). Letting the SAC1 algorithm use \( m_T \) number of samples in every iteration, \( \lambda = 0.5 \) and \( \eta = 0.5 \), we have

\[
\text{Pr}_k \geq \frac{1}{2 \log \frac{1}{\sqrt{\alpha}}} \sum_{t=5}^{\log \frac{1}{\sqrt{\alpha}}} \left( \frac{D_{\alpha^*} (1 - Q_{D_{\alpha^*}})}{|D_{\alpha^*}| + Q_{D_{\alpha^*}}} \right).
\]
log \frac{\log 1}{\alpha} + n \log \frac{1}{\alpha^2}\right)\right]

as the \text{max} is upper bounded by plus.

We observe from the proof that the non-convexity can result in non-zero training error for the learning algorithms in SAC algorithms, and thus the search process is interfered. But as long as the non-convexity is not quite severe, like the Spike Functions, SAC algorithms are not significantly affected, and can still be better than the uniform search by a factor near \(\frac{1}{\sqrt{\alpha}} / \log \frac{1}{\alpha}\).

**B. A Super-Polynomial Acceleration Condition**

We have shown in Proposition 4 that SAC algorithms using common classification algorithms cannot super-polynomially improve from the uniform search in the worst case. An interesting question is therefore raised that when the super-polynomial improvement is possible.

Learned from the proof of Proposition 4, a straightforward way is to use a powerful classification algorithm with exponentially improved sample complexity, i.e., \(O(\log \frac{1}{\delta})\), so that only a polynomial number of samples is required to achieve a super-polynomially small error. Several \textit{active learning} algorithms can do this in some circumstances (e.g. [7, 49]). Applying active learning algorithms needs a small modification of SAC.

In iteration \(t\), instead of sampling from the uniform distribution in \(D_{h_t}\), the sampling is guided by the classifier. Nevertheless, the achieved error is still evaluated under the original (uniform) distribution. Using such learning algorithms denoted as \(\mathcal{L}_{\text{sphere}}\), we achieve Proposition 4 showing a super-polynomial acceleration from the uniform search on Sphere Functions.

**Proposition 4** For any function in \(\mathcal{F}_{\text{sphere}}^n\) and any approximation level \(\alpha > 0\), SAC algorithms using \(\mathcal{L}_{\text{sphere}}\) can achieve the PAA query complexity, for any \(n \geq 2\),

\[
O\left(\log \frac{1}{\alpha} + \frac{1}{\delta} + n \log \frac{1}{\alpha^2}\right)
\]

with probability at least \(1 - \delta\).

**Proof.** We choose \(\alpha_t = \frac{1}{\sqrt{T}}\) for all \(t\), and use the number of iterations \(T\) to approach \(|D_{\text{true}}| = |D_{\alpha}|\), for the approximation level \(\alpha^*\). Solving this equation with the sphere volume results in \(T = \log \frac{(C_n)^2}{\alpha^2}\). We let the SAC algorithm run \(T = \log \frac{1}{\alpha^2}\) number of iterations. We assume \(\log \frac{1}{\alpha^2}\) is an integer for simplicity, which does not affect the generality.

In iteration \(t\), using \(\mathcal{L}_{\text{sphere}}\), we want the error of the hypothesis \(h_t, \epsilon_{D_{h_t}}\) to be \(\frac{1}{\alpha^{t+1}}\). Since the \(\mathcal{L}_{\text{sphere}}\) has the sample complexity \(O(\log \frac{1}{\delta})\), we ask for a hypothesis with zero training error, which requires the sample size \(m_t = O(tn) = O(n \log \frac{1}{\alpha^2})\) with \(\eta\) being a constant.

We thus obtain \(\sum_{t=1}^T m_t = O\left(n \left(\log \frac{1}{\alpha^2}\right)^2\right)\).

Following Eq. (1), we use uniform sampling within \(D_{h_t}\), then \(Q = \frac{1}{\alpha^2}\). Letting the SAC algorithms use \(m_{\text{true}}\) number of samples in every iteration, \(\lambda = 0.5\) and \(\eta = 0.5\), we have

\[
\overline{\mathcal{P}}_{h_t} \geq \frac{1}{2\log \frac{1}{\alpha}} \sum_{t=1}^T \log \frac{1}{\alpha} \left(\left|D_{\alpha^t}\right| (1 - Q\epsilon_{D_{h_t}}) + Q\epsilon_{D_{h_t}}\right)
\]

\[
\geq \frac{C_n(\alpha^*)^2}{2\log \frac{1}{\alpha}} \sum_{t=1}^T \log \frac{1}{\alpha} \left(1 - \frac{2}{2\log \frac{1}{\alpha}} \right)^2
\]

\[
\geq \frac{C_n(\alpha^*)^2}{2\log \frac{1}{\alpha}} \sqrt{2(C_n + 2) + 2(1/2)^2}
\]

\[
\geq \frac{C_n(\alpha^*)^2}{2\log \frac{1}{\alpha}} \left(\frac{1}{\alpha^2}\right)^2 (1 - (2\alpha^*)^2) = \Omega\left(\frac{1}{\alpha^2}\right).
\]

So we obtain the query complexity from Theorem 1, letting \(m_0\) be a constant,

\[
O \left(\max \left\{ \log \frac{1}{\alpha} \ln \frac{1}{\delta}, n \log \left(\frac{1}{\alpha^2}\right)\right\} \right)
\]

which is \(O\left(\frac{1}{\sqrt{\alpha}} (\ln \frac{1}{\delta} + n\log \frac{1}{\alpha^2})\right)\).

Meanwhile, we are more interested in exploring conditions under which the super-polynomial improvement is possible without requiring such powerful learning algorithms. For this purpose, we find the \textit{one-side-error} condition.

**Condition 2** (One-Side-Error) In SAC algorithms, for any \(t\) and any \(x \in X\), if \(x \in D_{h_t} \Delta D_{\alpha^t}\), it must hold that \(x \in D_{\alpha^t}\).

The condition implies that \(h_t\) can only make false-negative errors, i.e., wrongly classifies positive samples (inside \(D_{\alpha^t}\)) as negative, but no false-positive errors. One practical way to approach this condition is through the cost-sensitive classifiers ([14, 46]) with a very large mis-classification cost for negative samples. We call SAC algorithms that are further under this condition as SAC algorithms.

**Lemma 5** For SAC algorithms, it holds for all \(t\) that

\[|D_{h_t}| \leq |D_{\alpha^t}|.\]

**Proof.** Note that for training \(h_t\) we label the samples from \(D_{\alpha^t}\) as positive and label the rest as negative. Since \(h_t\) only makes false-negative errors, i.e., every error is in \(D_{\alpha^t}\), we have \(D_{h_t} \subseteq D_{\alpha^t}\), which implies the lemma.

**Lemma 6** shows that the one-side-error condition controls the size \(|D_{h_t}|\) to be bounded by \(|D_{\alpha^t}|\). Thus we can refine Lemma 3 as Lemma 6.

**Lemma 6** For SAC algorithms, it holds for all \(t\) that

\[|D_{\alpha^t} \cap D_{h_t}| \geq |D_{\alpha^t}| \left(1 - \epsilon_{D_{h_t}}\right)\]

where \(\epsilon_{D_{h_t}}\) is the expected error rate of \(h_t\) under \(U_X\).

**Proof.** Since the SAC algorithm is also a SAC algorithm, incorporating Lemma 5 into Lemma 3 proves the lemma.

We assume that \(\mathcal{L}_{\text{sphere}}\) is a learning algorithm that not only behaviors like \(\mathcal{L}_{\text{sphere}}\) but also results a hypothesis...
Proof. By Lemma \ref{lem:condition}, of iterations. We assume that we only ask for a random guess classification (i.e., error
the complexity of the uniform search. It is interesting to note
which is \(O(1/\alpha)\), and use the number of iterations
\(T\) to achieve \(|D_{\alpha^*}^*| = |D_{\alpha^*}|\), for the approximation level \(\alpha^*\). Solving this equation with the sphere volume results in \(T = \frac{\log (C_\alpha^*) \delta^\alpha}{\alpha^*}\). We let the SAC algorithm run \(T = \frac{\log \frac{1}{\alpha^*}}{\alpha^*}\) number of iterations. We assume \(\log \frac{1}{\alpha^*}\) is an integer for simplicity, which does not affect the generality.

In iteration \(t\), using \(L_{\text{sphere}}^+\), we want the error of the hypothesis \(h_t, \epsilon_{D_t}\), to be a constant \(\frac{1}{2}\). Since \(L_{\text{sphere}}^+\) produces a hypothesis with zero training error, to achieve \(\epsilon_{D_t} \leq \frac{1}{2}\) it requires the number of samples in \(O(n)\). We thus obtain \(\sum_{t=1}^{T} m_t = O \left( n \log \frac{1}{\alpha^*} \right)\).

We then follow Eq.\ref{eq:prop}. We use uniform sampling within \(D_{h_t}\), then \(Q = \frac{1}{\alpha^*}\). Letting the SAC algorithm use \(m_T\) number of samples in every iteration, \(\lambda = \frac{1}{3}\) and \(\eta = 0.5\), we have

\[
\Pr_h \geq \frac{1}{2 \log \frac{1}{\alpha^*}} \sum_{t=1}^{T} \log \frac{1}{\alpha^*} \left( \frac{|D_{\alpha^*}^*|}{|D_{\alpha^*}|} \right) \geq \frac{1}{2 \log \frac{1}{\alpha^*}} \sum_{t=1}^{T} \log \frac{1}{\alpha^*} \left( \frac{1}{|D_{\alpha^*}|} \right) = \frac{C_n(\alpha^*)^{\frac{1}{2}}}{8 \log \frac{1}{\alpha^*} \left( \frac{1}{|D_{\alpha^*}|} \right)} \geq \frac{C_n(\alpha^*)^{\frac{1}{2}} \left( \left( \frac{1}{\alpha^*} \right)^{\frac{1}{2}} - 1 \right)}{\frac{C_n}{\alpha^*}} = \Omega\left( \frac{1}{\log \frac{1}{\alpha^*}} \right).
\]

So we obtain from Theorem \ref{thm:complexity} the query complexity of the SAC algorithm, letting \(m_0\) be a constant,

\[
O \left( \max \left\{ \log \frac{1}{\alpha^*} \ln n, \frac{1}{\delta^*} \frac{1}{n} \right\} \right),
\]

which is \(O \left( \frac{1}{\alpha^*} \ln \frac{1}{\delta} \ln \frac{1}{\alpha^*} \right)\).

Proposition \ref{prop:improve} shows a super-polynomial improvement from the complexity of the uniform search. It is interesting to note that we only ask for a random guess classification (i.e., error rate \(\frac{1}{2}\)) in the proof of Proposition \ref{prop:improve}.

**Proposition 5** For any function in \(F_{\text{sphere}}^n\) and any approximation level \(\alpha^* > 0\), SAC algorithms can achieve the PAA query complexity

\[
O \left( \log \frac{1}{\alpha^*} \ln \frac{1}{\delta} n \log \frac{1}{\alpha^*} \right),
\]

with probability at least \(1 - \delta\).

**Proof.** By Lemma \ref{lem:condition}, we have

\[
\frac{|D_{\alpha^*}^*} \cap D_{h_t}| \geq \frac{|D_{\alpha^*}^*| \left( 1 - Q \epsilon_{D_t} \right)}{|D_{\alpha^*}|},
\]

where \(\epsilon_{D_t}\) is the error of \(h_t\) under its original distribution \(D_t\), and \(Q\) is the resulting factor of changing the distribution.

Let \(\alpha_t = \frac{1}{\alpha^*}\) for all \(t\), and use the number of iterations \(T\) to achieve \(|D_{\alpha^*}^*| = |D_{\alpha^*}|\), for the approximation level \(\alpha^*\). We thus obtain \(\sum_{t=1}^{T} m_t = O \left( n \log \frac{1}{\alpha^*} \right)\).

In the case study on the Sphere Function class, we find that a recent work \cite{4} also noticed that a learnable algorithm may not be necessary for a good SAC algorithm. It is interesting that a recent work \cite{4} also noticed that a learnable concept is not necessary for the trial-and-error search with a computation oracle.

In this paper, the SAC algorithms are analyzed in continuous domains, while the main body of theoretical studies of evolutionary algorithms focuses on the discrete domains. Thus understanding the performance of SAC algorithms in discrete domains is our future work. Moreover, in the SAC algorithms analyzed in this paper, the learning algorithm does not utilize the last hypothesis or the last data set. It would be interesting to investigate whether considering them will bring any significant difference.

**ACKNOWLEDGEMENTS**

This work was supported by the National Science Foundation of China (61375061) and the Jiangsu Science Foundation (BK2012303).

**REFERENCES**
