Revisiting the Cross Entropy Method with Applications in Stochastic Global Optimization and Reinforcement Learning

Ajin George Joseph\textsuperscript{1} and Shalabh Bhatnagar\textsuperscript{2}

Abstract. In this paper, we provide a new algorithm for the problem of stochastic global optimization where only noisy versions of the objective function are available. The algorithm is inspired by the well known cross entropy (CE) method. The algorithm takes the shape of a multi-timescale stochastic approximation algorithm, where we reuse the previous samples based on discounted averaging, and hence it saves the overall computational and storage cost. We provide proof of the stability and the global optimization property of our algorithm. The algorithm shows good performance on the noisy versions of global optimization benchmarks and outperforms a state-of-the-art algorithm for non-linear function approximation in reinforcement learning.

1 Introduction and Preliminaries

In this paper, we solve the following problem: For a latent probability measure $P_y$ over $Y \subset \mathbb{R}^n$,
\[
\text{Find } x^* \in \arg \max_{x \in X} \mathbb{E}_y [J(y) , x],
\]
where $L : \mathbb{R}^k \times \mathbb{R}^m \rightarrow \mathbb{R}$ is a bounded continuous function and $J : Y \rightarrow \mathbb{R}^k$ is a bounded function (hence $\mathbb{E}_y [[J]] < \infty$). Here $\mathbb{E}_y [.]$ is the expectation w.r.t. $P_y$. The region $X$ referred as the solution space is a compact subset of $\mathbb{R}^m$. For brevity, we let $H(x) \triangleq L(\mathbb{E}_y [J(y) , x])$. In this paper, we assume $x^*$ is unique and $x^* \in \text{interior}(X)$.

In this paper, we consider a “black-box” setting, where neither a closed form expression nor structural properties of the objective function $H$ are available. However, for a given $x \in X$, a noisy measurement $\hat{H}(x)$ of the objective function $H$ is available, where $\hat{H}(x) = H(x) + \epsilon(x)$. Here $\epsilon(x)$ is the noise incurred during the measurement of $H(x)$ which is primarily attributed to the inability to measure accurately the quantity $\mathbb{E}_y [\cdot]$. The intractability of $\mathbb{E}_y [\cdot]$ is due to the hidden probability measure $P_y$. Problems of the above kind are found in areas of engineering and in discrete-event system simulation [11, 13].

Intuitively, it follows directly from the definition of the problem that any algorithm which solves the problem of the above kind has to be a zero-order method. Since the higher order values of the objective function are unavailable, the algorithm relies only on the function values. These methods are generally called gradient free methods. A few prominent algorithms of this kind are simultaneous perturbation stochastic approximation method (SPSA) [26], model reference adaptive search method (MRAS) [8], cross entropy method (CE) [25], estimation of distribution algorithms (EDA) [29] and gradient based adaptive search (GASSO) [30]. The advantage of the above methods is the non-dependency on the structural properties of the objective function and hence can be applied in a generalized setting.

In this paper, we consider the well known cross entropy method. The cross entropy method was motivated by the rare event probability estimation method proposed in [22], which is a variance reduction technique. Later, an iterative procedure based on the above method was applied in various combinatorial optimization problems [23]. Cross entropy method also found successful applications in continuous multi-extremal optimization [21]. A few other applications of the CE method include buffer allocation [1], queuing models [3], DNA sequence alignment [14], control and navigation [6], reinforcement learning [17, 19] and several NP-hard problems [24, 21].

In this paper, we apply a stochastic approximation variant of the CE method to the continuous multi-extremal stochastic optimization problem defined in (1). This particular version of the algorithm is designed to overcome the drawbacks arising due to the inordinate computational and storage requirements of the original CE method. The stochastic approximation nature of our algorithm also helps to adapt naturally to the stochastic optimization setting.

2 The Cross Entropy Method

Notation: We use $x$ for random variable and $x$ for deterministic variable. Let $\lceil a \rceil$ denote the smallest integer greater than $a$. Let $\text{supp}(f) \triangleq \{x | f(x) \neq 0\}$ and $\text{interior}(A)$ be the interior of set $A$. Let $f_\theta(\cdot)$ denote the probability density function parametrized by $\theta$ and $\mathbb{E}_\theta [\cdot]$ be the expectation w.r.t. $f_\theta$. Let $\gamma_\rho(\cdot, \theta)$ denote the $(1 - \rho)$-quantile of $\mathcal{H}(x)$ w.r.t. $f_\theta$, i.e.,
\[
\gamma_\rho(\cdot, \theta) \triangleq \sup \{l : \mathbb{E}_\theta [\mathcal{H}(x) \geq l] \geq \rho \}. \tag{2}
\]
For $x, y \in \mathbb{R}$, define $\chi(x, y) = \begin{cases} 1 & \text{if } x \geq y \\ 0 & \text{otherwise.} \end{cases} \tag{3}$

The Cross Entropy (CE) method [25, 9, 4] is a zero-order optimization method to solve optimization problems where the objective function does not possess smooth differentiable structural properties. CE method belongs to a broader class of methods called the model based search methods. The goal of the CE method is to find an optimal “model” or probability distribution over the solution space $X$ which concentrates on the global maxima of the objective function. The CE method is an iterative procedure where at each iteration $k$, a search is conducted on a space of parametrized probability distributions $\{f_\theta | \theta \in \Theta\}$ on $X$, where $\Theta$ is the parameter space, to find a

\textsuperscript{1} Indian Institute of Science, Bangalore, email: ajin@csa.iisc.ernet.in
\textsuperscript{2} Indian Institute of Science, Bangalore, email: shalabh@csa.iisc.ernet.in
distribution parameter $\theta_k$ which reduces the Kullback-Leibler (KL) distance from the optimal model. The most commonly used distribution family here is the exponential family of distributions.

**Exponential Family of Distributions:** These distributions are represented by $C \triangleq \{f_\theta(x) = h(x)e^{\theta^T\Gamma(x)} - K(\theta) \mid \theta \in \Theta \subset \mathbb{R}^d\}$, where $h : \mathbb{R}^m \rightarrow \mathbb{R}$, $\Gamma : \mathbb{R}^m \rightarrow \mathbb{R}^d$, and $K : \mathbb{R}^d \rightarrow \mathbb{R}$ and $K(\theta) = \log \int h(x)e^{\theta^T\Gamma(x)}dx$. The Gaussian distribution with mean vector $\mu \in \mathbb{R}^m$ and the covariance matrix $\Sigma \in \mathbb{R}^{m \times m}$ belongs to $C$. In this case,

$$f_\theta(x) = \frac{1}{\sqrt{(2\pi)^m|\Sigma|}} e^{-(x-\mu)^T\Sigma^{-1}(x-\mu)/2},$$

and so one may let $h(x) = 1/(2\pi)^{m/2}$, $\Gamma(x) = (x,x)^T$ and $\theta = (\Sigma^{-1}\mu, -\frac{1}{2}\Sigma^{-1})^T$.

**Assumption (A1):** The parameter space $\Theta$ is a compact set.

### 2.1 CE Method (Ideal Version)

The CE method aims to find a sequence of model parameters $\{\theta_k \in \Theta \}_{k \in \mathbb{Z}_+}$ and an increasing sequence of thresholds $\{\gamma_k \in \mathbb{R}_+ \}_{k \in \mathbb{Z}_+}$ with the property that the support of $f_\theta$ satisfies $\text{supp}(f_\theta) \subseteq \{x \mid H(x) \geq \gamma_k\}$.

**Threshold Evaluation:**

$$\gamma_{k+1} = \arg \min_{\gamma \in [\gamma_k, \infty)} \mathbb{E}_{\theta_k}[\varphi(H(x))\chi(H(x), \gamma)]J(x),$$

where $\varphi(r) = \gamma \exp(-\gamma r)$ for $r \in \mathbb{R}_+$.

**Optimization Problem:**

$$\theta_{k+1} = \arg \max_{\theta \in \Theta} \mathbb{E}_{\theta}[\varphi(H(x))\chi(H(x), \gamma_{k+1})]f_\theta(x).$$

**Algorithm:**

**Algorithm 1:** The Monte-Carlo CE Algorithm

1. **Sampling Candidate Solutions:** Sample $N_k$ i.i.d. solutions $\Lambda_k = \{x_1, \ldots, x_{N_k}\}$ using $f_\theta$.

2. **Performance Evaluation:** For each $x$ in $\Lambda_k$, take $M_k$ observations $\{y_{i,k}\}_{i=1}^{M_k}$, where $y_i \sim P_Y$ and calculate the sample average $\bar{H}(x) = \frac{1}{N_k} \sum_{i=1}^{N_k} J(y_{i,k}, x)$. Also, use $\gamma_{k+1}$.

3. **Threshold Evaluation:** Calculate the sample $(1-\rho)$-quantile $\gamma_{k+1} = \bar{H}_{\gamma_{k+1}}(\{x\}_{i=1}^{N_k})$, where $\bar{H}_{\gamma} = \frac{1}{N_k} \sum_{i=1}^{N_k} J(y_{i,k}, x)$.

4. **Threshold Comparison:**

   - If $\gamma_{k+1} \geq \gamma^*_k + \epsilon$ then $\gamma_{k+1} = \gamma_{k+1}$; else $\gamma_{k+1} = \gamma_{k+1}^*$.

5. **Model Parameter Update:** Update $\theta_{k+1} = (\mu_{k+1}, \Sigma_{k+1})^T$ using (11) and (11) with $\gamma_{k+1}$ instead of $\gamma_{k}^*$.

6. If the stopping rule is satisfied, then return $\theta_{k+1}$ and terminate, else set $k := k + 1$ and go Step 1.
2.3 Drawbacks of the CE Method

- Inefficient use of prior information: The naive approach of the Monte-Carlo CE does not utilize prior information efficiently. Note that Monte-Carlo CE possesses a stateless behaviour. At each iteration \( k \), a completely new collection of samples are drawn using the distribution \( f_{\theta_k} \). The samples are used to derive the estimates \( \gamma_{k+1}, \mu_{k+1} \) and \( \Sigma_{k+1} \). The algorithm does not utilize the estimates generated prior to \( k \).

- Computational limitations: These arise due to the dependence on the sample size \( N_k \). One does not know a priori the best value for the sample size \( N_k \). Higher values of \( N_k \) while resulting in higher accuracy also require more computational resources. One often needs to apply brute force in order to obtain a good choice of \( N_k \). Also as \( m \), the dimension of the solution space \( \mathcal{X} \), takes large values, more samples are required for better accuracy, making \( N_k \) large as well. This makes finding the \( \ell \)-th order statistic \( \hat{H}_i(\ell) \) in Step 3 harder. Note that the order statistic \( \hat{H}_i(\ell) \) is obtained by sorting the list \( \{\hat{H}(x_1), \hat{H}(x_2), \ldots, \hat{H}(x_{N_k})\} \). The computational effort required in that case is \( O(N_k \log N_k) \) which is expensive. Also note that \( N_k \) diverges to infinity and hence this super linear relationship is computationally very costly.

- Storage limitations: The storage requirement for sampling a set \( \mathcal{X} \) is \( N_k \times \text{size}(\mathcal{X}_i) \). In situations when \( m \) and \( N_k \) are large, the storage requirement is a major concern.

Note that similar issues are expected to arise with the sample size \( M_k \) also.

An illustration in Figure 1 demonstrates the dependency of the performance of Monte-Carlo CE on the sample size schedules \( \{N_k\} \) and \( \{M_k\} \). We consider here the Griewank function on \( \mathbb{R}^{100} \), cf. (36). We take \( M_{k+1} = \lceil \alpha_1 M_k \rceil \), \( \alpha_1 > 1 \) and \( N_{k+1} = \lceil \alpha_2 N_k \rceil \), \( \alpha_2 > 1 \). So a particular schedule can be identified by \( M_0, \alpha_1 \) and \( N_0, \alpha_2 \). Here we take \( \alpha_1 = 1.01 \) and \( \alpha_2 = 1.005 \) for all the schedules, however they differ in their initial values \( M_0 \) and \( N_0 \). From the Figure 1, note that for \( (M_0, N_0) \in \{(100, 100), (150, 150), (200, 200)\} \), the convergence behaviour is close, however for \( (M_0, N_0) \in \{(500, 500), (1000, 1000), (2000, 2000)\} \), Monte-Carlo CE converges to a better value. Also when observed carefully, we can notice a significant difference in the limit point of each individual trajectory.

Different variants of the naive Monte-Carlo CE have been proposed in the literature, such as the gradient based Monte-Carlo cross entropy method (GMCCCE) [10] and the modified Monte-Carlo cross entropy method [28]. All the variants differ only in the model updating step, the other steps remain the same. Hence they also suffer from the above drawbacks.

3 Proposed Algorithm

In this paper, we resolve the shortcomings of the Monte-Carlo CE method with regards to the concerns mentioned in the previous section, by remodelling the same in the stochastic approximation framework. We follow the same sequence of steps as in Algorithm 1, but differ in their implementation. The stochastic approximation approach streamlines the batch processing of the Monte-Carlo CE. We provide a multi-timescale stochastic approximation algorithm which is efficient, stable, incremental in nature and imposes minimal restriction on the objective function. We avail ourselves of the continuity relationship that holds between the \((1 - \rho)\)-quantile \( \gamma_\rho(H(\cdot), \theta) \) and the model parameter \( \theta \). We also exploit the continuity that holds between \( \mu_{k+1} \) and \( \Sigma_{k+1} \) w.r.t. \( \theta_k \). The bootstrapping property inherent in the stochastic approximation algorithms helps to utilize these relationships efficiently.

Stochastic approximation algorithms [2, 15, 20] are a natural way of utilizing prior information. It does so by discounted averaging of the prior information and are usually expressed as recursive equations of the following form:

\[
z_{k+1} = z_k + \alpha_{k+1} \Delta z(z_k, b_k, D_{k+1}),
\]

where \( \Delta z(z, b, D) = h(z) + b + D \) is the increment term, \( b_k \) is the bias term with \( b_k \to 0 \), \( D_k \) is a random noise with zero-mean and \( h(\cdot) \) a Lipschitz continuous function. The learning rate \( \alpha_k \) satisfies \( \Sigma \alpha_k = \infty, \Sigma \alpha_k^2 < \infty \).

In our algorithm, we do not apply the naive order statistic to estimate the \((1 - \rho)\)-quantile. Rather we employ a stochastic recursion which serves the same purpose, but efficiently. To achieve this, we make use of the following lemma from [7]. This lemma provides a characterization of the quantile in terms of an optimization problem.

**Lemma 1** The \((1 - \rho)\)-quantile of a bounded real-valued function \( H(\cdot) \) (with \( H(x) \in [H_1, H_2] \)) w.r.t the probability distribution \( f_\omega(\cdot) \) is reformulated as the optimization problem

\[
\text{Find } \gamma_\rho(H(\cdot), \theta) = \min_{u \in [H_1, H_2]} \mathbb{E}_\theta[\psi(H(x), u)],
\]

where \( \psi(H(x), u) = (1 - \rho)(H(x) - u)(H(x), u) + \rho(u - H(x))\chi(H(x)). \)

We develop a stochastic gradient recursion which solves the optimization problem in (13). The increment term for the recursion is the sub-differential of \( \psi \) w.r.t. \( u \), and is given by

\[
\nabla_u \psi(H(x), u) = -(1 - \rho)\chi(H(x), u) + \rho\chi(u, H(x)).
\]

The model parameter update step involves three stochastic recursions. For this we introduce two new variables, \( \xi^{(0)}(\cdot) \) and \( \xi^{(1)}(\cdot) \), which estimate the RHS of equations (6) and (7) respectively. We also require two increment functions which are defined as follows:

\[
\Delta \xi^{(0)}(x, \omega, \gamma) = g_1(L(\omega, x), x, \gamma) - \xi^{(0)}_0(L(\omega, x), \gamma),
\]

\[
\Delta \xi^{(1)}(x, \omega, \mu, \gamma) = g_2(L(\omega, x), x, \gamma, \mu) - \xi^{(1)}_0(L(\omega, x), \gamma).
\]

The algorithm is formally presented in Algorithm 2.
Note that the algorithm uses only 3 samples per iteration: $y_{k+1}, x_{k+1}$ and $x_{k+1}^{old}$. This is a big improvement both computational and storage wise, compared to the original CE method.

Mixture Distribution: In the algorithm, we use a mixture distribution $f_{\theta_k}$ to generate the sample $x_{k+1}$, where $f_{\theta_k} = (1-\lambda)f_{\theta_{old}} + \lambda f_{\theta_0}$ with $\lambda$ the mixing weight. $\lambda$ takes its values from a pre-defined decaying sequence $\{\lambda_k\}_{k \in \mathbb{Z}_+}$, with assignment happening in (24) during the model parameter update step. The initial parameter $\theta_0$ is chosen s.t. the density function $f_{\theta_0}$ is strictly positive on every point in the solution space $\mathcal{X}$, i.e., $f_{\theta_0}(x) > 0, \forall x \in \mathcal{X}$. The mixture approach facilitates exploration of the solution space and prevents the iterates from getting stranded in suboptimal solutions.

Learning Rates: The learning rates $\alpha_k, \beta_k$ and the mixing weight $\lambda_k$ are deterministic, non-increasing and satisfy the following:

$$\alpha_k > 0, \beta_k > 0, \beta_k > 0, \lim_{k \to \infty} \alpha_k = 0, \lim_{k \to \infty} \beta_k = 0, \lim_{k \to \infty} \lambda_k = 0,$$

$$\sum_{k=1}^{\infty} \alpha_k = \infty, \sum_{k=1}^{\infty} \beta_k = \infty, \sum_{k=1}^{\infty} (\alpha_k^2 + \beta_k^2) < \infty, \lim_{k \to \infty} \frac{\alpha_k}{\beta_k} = 0. \quad (25)$$

Since $\alpha_k \to 0$ faster than $\beta_k$, the timescale obtained from $\beta_k, k \geq 0$ is faster as compared to the other.

Threshold Comparison Step: The threshold comparison is achieved using the recursion (22) of the random variable $T_k$. The model parameter $\theta_k$ is not updated at each $k$. Rather it is updated whenever $T_k$ arches over $\epsilon_1$, where $\epsilon_1 \in (0,1)$ is a constant. So the update of $\theta_k$ only happens along a subsequence $\{k(n)\}_{n \in \mathbb{Z}_+}$ of $\{k\}_{k \in \mathbb{Z}_+}$. Between $k = k(n)$ and $k = k(n+1)$, the variable $\gamma_k$ estimates $(1-\rho)$-quantile of $L(\omega_k, \cdot)$ w.r.t. $f_{\theta_{k(n)}}$. The threshold $\gamma_k^*$ is also updated in (24) during the $\epsilon_1$ crossover. Thus $\gamma_k^* = \hat{\gamma}_k(n)$ is the estimate of $(1-\rho)$-quantile w.r.t. $f_{\theta_{k(n-1)}}$.

Notation: We denote by $\gamma_{\rho}(L(\omega, \cdot), \hat{\theta})$, the $(1-\rho)$-quantile of $L(\omega, \cdot)$ w.r.t. the mixture distribution $\hat{f}_{\theta}$ and let $\hat{E}_{\rho}[]$ be the expectation w.r.t. $\hat{f}_{\theta}$.

Proposition 1: $T_k$ belongs to $(1, \infty)$, $\forall k > 0$.

Proof: By rearranging (22) we get,

$$T_{k+1} = (1-c)T_k + c(\chi(\gamma_{k+1}, \gamma_{k+1}^*) - \chi(\gamma_{k+1}, \gamma_{k+1})), \quad (22)$$

where $c \in (0,1)$. In the worst case, either $\chi(\gamma_{k+1}, \gamma_{k+1}^*) = 1, \forall k$ or $\chi(\gamma_{k+1}, \gamma_{k+1}^*) = 1, \forall k$. Since the two events are mutually exclusive, we will only consider the former event $\{\chi(\gamma_{k+1}, \gamma_{k+1}^*) = 1, \forall k\}$. In this case

$$\lim_{k \to \infty} T_k = \lim_{k \to \infty} (c + c(1-c) + \cdots + c(1-c)^{k-1}) = 1.$$

Similarly for $\{\chi(\gamma_{k+1}, \gamma_{k+1}^*) = 1, \forall k\}$, we have $T_k \to -1$.

$T_k$ in (22) serves two purposes: first, it is a delay mechanism, whereby it stalls the $\theta_k$-recursion so that $\xi_{k+1}$ and $\xi_{k+1}^*$ are sufficiently close to their true values. Second, it ensures that the estimates $\gamma_k$ eventually become greater than the current threshold $\gamma_k(n)$ i.e., $\gamma_k > \gamma_k^*(n)$ for all but finitely many $k$.

Remark 2: The recursion (21) is not addressed in the discussion above. The assignment of $\gamma_k^{(n)}$ in (24) happens along a subsequence $\{k(n)\}_{n \geq 0}$. Hence $\gamma_k(n)$ is the estimate of $\gamma_{\rho}(L(\omega_k, \cdot), \theta_{k(n-1)})$. But at time $k(n) < k \leq k(n+1)$, $\gamma_k(n)$ is compared with $\gamma_k(n)$ in (22). But $\gamma_k$ is derived from a better estimate of $L(\omega, \cdot)$, Equation (21) ensures that $\gamma_k(n)$ is updated using the latest estimate of $L(\omega, \cdot)$. The variable $\theta_{k(n-1)}$ holds the model parameter $\theta_{k(n-1)}$ and the update of $\gamma_k(n)$ in (21) is performed using $x_{k(n)+1}$ sampled from $f_{\theta_{k(n)}}$.

3.1 Convergence Analysis

Assumption (A2): The sequences $\{\omega_k\}_{k \in \mathbb{Z}_+}$ and $\{\gamma_k\}_{k \in \mathbb{Z}_+}$ in (17) and (18) resp. satisfy $\sup_k |\omega_k| < \infty$ and $\sup_k |\gamma_k| < \infty$ a.s.

Remark 3: The assumption (A2) is a technical requirement to prove convergence of the algorithm. A commonly used procedure to ensure almost sure boundedness of iterates in a stochastic iterative scheme is to project these after each update on to a priori chosen (large enough) compact and convex set. In this case, the bound on the compact set can be derived from the bound on $L(\cdot, \cdot)$. Note that the recursion (17) is independent of other recursions and
hence can be analysed independently. For the recursion (17) we have the following result.

**Lemma 2** Let the step-sizes $\alpha_k$ and $\beta_k$, $k \in \mathbb{Z}_+$ satisfy (25). Also let (A2) hold. Then the iterates $\omega_k$ in (17) satisfy $\omega_k \to \omega^*$ = $\mathbb{E}_\theta [J(y)]$ as $k \to \infty$ w.p. 1.

As noted above, the update of $\theta_k$ only happens along a subsequence $\{k_i(n)\}_{n \in \mathbb{Z}_+}$ of $\{k\}_{k \in \mathbb{Z}_+}$, of $\{k\}_{k \in \mathbb{Z}_+}$. So between $k = k_i(n)$ and $k = k_i(n+1)$, $\theta_k$ is constant. The lemma and the theorem that follow in this paper depend on the timescale difference in the step-size schedules $\{\alpha_k\}_{k \in \mathbb{Z}_+}$ and $\{\beta_k\}_{k \in \mathbb{Z}_+}$. The step-size $\{\beta_k\}_{k \in \mathbb{Z}_+}$ decays to 0 at a slower rate than $\{\alpha_k\}_{k \in \mathbb{Z}_+}$, and hence the increments in the recursions (18), (19) and (20) which are controlled by $\beta_k$ are larger and hence converge faster than the recursions (17) and (23) which are controlled by $\alpha_k$. So the relative evolution of the variables from the slower timescale $\alpha_k$ i.e. $\omega_k$, $\theta_k$ is slow and can be considered constant when viewed from the faster timescale $\beta_k$. See Chapter 6, [2] for a description on multi-timescale algorithms.

Hence, when viewed from the timescale of the recursion (18), one may consider $\omega_k$ and $\theta_k$ to be fixed. For recursion (18) we have the following result:

**Lemma 3** Let $L(\omega_k, \cdot) \equiv L(\omega, \cdot)$, $\theta_k \equiv \theta$, $\forall k$. Also let (A2) hold. Then $\gamma_k$ in (18) satisfy $\lim_{k \to \infty} \gamma_k = \gamma(\rho(\omega, \cdot), \bar{\theta})$ w.p. 1.

Again, when viewed from the timescales of the recursions (19) and (20), one may consider $\omega_k$ and $\theta_k$ to be fixed as before. For the recursions (19) and (20), we have the following:

**Lemma 4** Assume $L(\omega_k, \cdot) \equiv L(\omega, \cdot)$, $\theta_k \equiv \theta$, $\forall k$. Then a.s.,

(i) $\lim_{k \to \infty} \xi^{(0)}_{\omega_k} = \xi^{(0)}$ = $\mathbb{E}_{\bar{\theta}} \left[ \langle g_1 \left( L(\omega, x), \gamma_\rho(\omega), \bar{\theta} \right) \rangle \right]$ = $\mathbb{E}_{\bar{\theta}} \left[ \langle g_0 \left( L(\omega, x), \gamma_\rho(\omega), \bar{\theta} \right) \rangle \right]$

(ii) $\lim_{k \to \infty} \xi^{(1)}_{\omega_k} = \xi^{(1)} = \mathbb{E}_{\bar{\theta}} \left[ \langle g_2 \left( L(\omega, x), \gamma_\rho(\omega), \bar{\theta} \right) \rangle \ess{\xi^{(0)}_{\omega_k}} \right]$.

(iii) If $\gamma_\rho(\omega, \cdot, \bar{\theta}) > \gamma(\rho(\omega, \cdot), \bar{\theta}_{old})$, then $T_k$, $k \geq 0$ in (22) satisfy $\lim_{k \to \infty} T_k = 1$ a.s.

**Notation:** For the subsequence $\{k_i(n)\}_{n \geq 0}$ of $\{k\}_{k \in \mathbb{Z}_+}$, we denote $k_i(0) \equiv k_{i-1}$. Along the subsequence $\{k_i(n)\}_{n \geq 0}$ with $k_0 = 0$ the updating of $\theta_k$ can be expressed as follows:

$$\theta_{k_i(n+1)} = \theta_{k_i(n)} + \alpha_{k_i(n+1)} \Delta \theta_{k_i(n+1)},$$

where $\Delta \theta_{k_i(n+1)} = \langle \xi^{(0)}_{k_i(n+1)} \rangle - \theta_{k_i(n)}$. We will prove now that the increment term $\Delta \theta_{k_i(n+1)}$ in equation (26) is indeed an estimate of $\nabla \phi(\theta, \omega)$ at $\theta = \theta_{k_i(n)}$, where

$$\Psi(\theta, \omega) = \log \mathbb{E}_\theta \left[ \langle g_0 \left( L(\omega, x), \gamma_\rho(\omega, \cdot) \right) \rangle \right]$$

with $\Psi(\theta, \omega)$ = $\log \mathbb{E}_\theta \left[ \langle g_0 \left( L(\omega, x), \gamma_\rho(\omega, \cdot) \right) \rangle \right]$.

**Lemma 5** For the given function $L(\omega, \cdot) \in \mathbb{R}$, $\theta = (\mu, \Sigma)^T$ and $\theta(\cdot) = (\theta_1, \theta_2)^T = (\Sigma^{-1} \mu, -\frac{1}{2} \Sigma^{-1})$. Before proving this, we state a key lemma about the gradient of $\Psi$. We now state our main theorem. The following theorem guarantees the convergence of the model sequence $\{\theta_k\}_{k \in \mathbb{Z}_+}$ generated by Algorithm 2 and it further provides a characterization of its limit points. It also shows that by imposing additional structural restrictions on the objective function $\mathcal{H}$, the convergence of the algorithm to the degenerate distribution concentrated on the global maximum $x^*$ is ensured.

**Theorem 6** Let $\varphi(x) = \exp(rx)$, $r \in \mathbb{R}$. Assume that the objective function $\mathcal{H}$ satisfies the following two conditions: (i) $\nabla^2 \mathcal{H} \exists \mathbb{R}$ and (ii) $\phi^{\text{upper}}(x, \cdot)$ is continuous for $1 \leq i, j \leq m$. Let the learning rates $\alpha_k$ and $\beta_k$, $k \in \mathbb{Z}_+$ satisfy (25). Let $\{\theta_k = (\mu_k, \Sigma_k)^T\}_{k \in \mathbb{Z}_+}$ be the sequence generated by Algorithm 2 and assume $\theta_k \in \text{Interior}(\Theta)$, $\forall k \in \mathbb{Z}_+$, Also, let (A1) and (A2) hold. Then

$$\lim_{k \to \infty} \theta_k = \lim_{k \to \infty} \left( \mu_k, \Sigma_k \right)^T = (x^*, 0_{m \times m})^T \text{ w.p. 1},$$

where $x^*$ is defined in (1).

**Proof:** Rewriting the equation (23) along the subsequence $\{k_i(n)\}_{n \in \mathbb{Z}_+}$ we have for $n \in \mathbb{Z}_+$,

$$\theta_{k_i(n+1)} = \theta_{k_i(n)} + \alpha_{k_i(n+1)} \langle \xi^{(0)}_{k_i(n+1)} \rangle - \theta_{k_i(n)}.$$

Also suppose $\|\theta_{k_i(n)}\| < \infty$ a.s. Rearranging the equation (28) we get, for $n \in \mathbb{Z}_+$,

$$\theta_{k_i(n+1)} = \theta_{k_i(n)} + \alpha_{k_i(n+1)} \left( \mathbb{E} \left[ \nabla \phi(\theta_{k_i(n)}, \omega^*) \right] \theta_{k_i(n)} \right) + o(1),$$

where the $o(1)$ term corresponds to errors in the estimation of $\xi^{(0)}_{k_i(n)}$ and $\xi^{(1)}_{k_i(n)}$ that decays to zero a.s. from Lemma 4.

Now consider the gradient flow ODE

$$\frac{d\theta(t)}{dt} = \nabla \phi(\theta(t), \omega^*), \quad t \in \mathbb{R}_+,$$

where $\omega^*$ is defined in Lemma 2.

By appealing to Theorem 2, Chapter 2 of [2], the asymptotic equivalence between the equations (29) and (30) can be easily established. Therefore, the recursion (23) reduces to a stochastic gradient ascent procedure which optimizes the objective function $\Psi(\theta, \omega^*)$. Hence the limiting behaviour of the model sequence $\{\theta_k\}_{k \in \mathbb{Z}_+}$ can be obtained by analysing the same of the above ODE. The fixed points of the ODE (30) can be obtained by equating $\nabla \Psi$ to 0. Equating $\nabla \Psi(\theta, \omega^*)$ to $0$, we get

$$\mu = \mathbb{E}_\theta \left[ g_1 \left( H(x), \gamma_\rho(\omega^*, \cdot) \right) \right] - \mathbb{E}_\theta \left[ g_0 \left( H(x), \gamma_\rho(\omega^*, \cdot) \right) \right].$$

Equating $\nabla \Psi(\theta, \omega^*)$ to 0, we get

$$\mathbb{E}_\theta \left[ g_2 \left( H(x), \gamma_\rho(\omega^*, \cdot) \right) \right] = \Sigma = 0.$$

To further simplify the notation, let $\gamma_\rho(\theta) = \gamma_\rho(\omega^*, \cdot, \theta)$. Also, for brevity let $S(\theta) = \mathbb{E}_\theta \left[ g_0 \left( H(x), \gamma_\rho(\theta) \right) \right]$ and $g_0(x, \theta) = \mathbb{E}_\theta \left[ g_0 \left( H(x), \gamma_\rho(\theta) \right) \right]$. Substituting the expression for $\mu$ from (31) in (32) and after further simplification we get,

$$(1/S(\theta)) \mathbb{E}_\theta \left[ g_0(x, \theta) xx^T \right] = -\mu^T - \Sigma = 0.$$
Now consider \( \| \cdot \|_\omega \) and by \( \theta \).

By the assumption to the contrary, we denote by \( \theta \in \Theta \) follows from the assumption \( \varphi(x) = \exp(rx) \), and \( G(x) = \rho(x) \).

We use their noisy versions to evaluate our algorithm. The function \( \varphi(\cdot) \) is chosen as \( \varphi(x) = \exp(rx) \), where \( r \in \mathbb{R}^k \). We use the performance of our algorithm with the original Monte-Carlo CE (MCCE), simultaneous perturbation stochastic approximation (SPSA) and the gradient based Monte-Carlo CE (GMCCCE) [10] which is a modified version of MCCE. We consider here the noise injected version of SPSA, which is shown to have global optimization properties [18].

### 4 Experimental Results

For empirical evaluation, we use two settings: (1) Global optimization benchmarks and (2) Nonlinear function approximation setting in reinforcement learning. In each setting, the results shown are averages over 10 independent sample trajectories obtained with the same initial distribution \( \theta_0 \).

#### 4.1 Levy function \([m = 50]\) Continued, Differentiable

\[ \mathcal{H}_1(x) = 0.1 + G_1(x)\|E[Y]\|_2 - E[Y]^\top E[Z], \]

where \( G_1(x) = -1 - \sin^2(\pi y_1) \)-vector. The function has global maximum at the point \( x_i = 1, 1 \leq i \leq 20 \) with value \( \mathcal{H}^*_1 = 0.142135 \).

#### 4.2 Qing function \([m = 20]\) Continuous, Differentiable, Separable, Scalable, Multimodal

\[ \mathcal{H}_2(x) = 10^{-3} \times G_2(x)\|E[Y]\|_2 - E[Y]^\top E[Z], \]

where \( G_2(x) = -\sum_{i=1}^m (x_i^2 - i)^2 \).

The function has global maximum at the point \( x_i = \sqrt{i}, \forall i, 1 \leq i \leq m \) with value \( \mathcal{H}^*_2 = -160.0 \).
(3) **Griewank function** \( [m = 20] \) [Continuous, Differentiable, Non-Separable, Scalable, Multimodal]

\[
\mathcal{H}_3(x) = G_3(x) \frac{\|E[Y]\|}{2 - E[Y]} E[Z],
\]

where \( G_3(x) = -1 - \frac{1}{4000} \sum_{i=1}^{m} x_i^2 + \prod_{i=1}^{m} \cos(x_i/\sqrt{2}). \) (36)

\( Y, Z \in \mathbb{R}^{20} \) with \( Y \sim \mathcal{N}([2.0, 2.0, \ldots, 2.0]^\top, 40 \times I_{20 \times 20}) \) and \( Z \sim \mathcal{N}([4.0, 4.0, \ldots, 4.0]^\top, 40 \times I_{20 \times 20}) \). The global maximum is \( \mathcal{H}_3 = -160.0 \) and is achieved at the origin.

![Figure 3: Qing Function](image)

(a) The trajectory of \( H_2(\mu_k) \). The dotted horizontal line is \( H_2^* \), x-axis is the time in secs relative to the start of the algorithm, y-axis is the function value.

(b) Plot of 3-dimensional \( G_2(\cdot) \)

![Figure 4: Griewank Function](image)

(a) The trajectory of \( H_3(\mu_k) \). The dotted horizontal line is \( H_3^* \), x-axis is the time in secs relative to the start of the algorithm, y-axis is the function value.

(b) Plot of 3-dimensional \( G_3(\cdot) \)

![Figure 5: Rosenbrock Function](image)

(a) The trajectory of \( H_4(\mu_k) \). The dotted horizontal line is \( H_4^* \), x-axis is the time in secs relative to the start of the algorithm, y-axis is the function value.

(b) Plot of 3-dimensional \( G_4(\cdot) \)

**Table 1**: The parameter values used in experiment 1

<table>
<thead>
<tr>
<th>( L(\cdot, \cdot) )</th>
<th>( \alpha_k )</th>
<th>( \beta_k )</th>
<th>( \lambda_k )</th>
<th>( c )</th>
<th>( \epsilon_1 )</th>
<th>( \rho )</th>
<th>( r )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( H_3 )</td>
<td>( \frac{1}{5} )</td>
<td>0.1</td>
<td>( k_{(19)}^{3, 0} )</td>
<td>0.06</td>
<td>0.9</td>
<td>0.1</td>
<td>0.01</td>
</tr>
<tr>
<td>( H_2 )</td>
<td>( \frac{1}{5} )</td>
<td>0.09</td>
<td>( k_{(19)}^{3, 0} )</td>
<td>0.06</td>
<td>0.9</td>
<td>0.01</td>
<td>0.002</td>
</tr>
<tr>
<td>( H_4 )</td>
<td>( \frac{1}{5} )</td>
<td>0.09</td>
<td>( k_{(19)}^{3, 0} )</td>
<td>0.06</td>
<td>0.9</td>
<td>0.01</td>
<td>0.001</td>
</tr>
<tr>
<td>( H_1 )</td>
<td>( \frac{1}{5} )</td>
<td>0.09</td>
<td>( k_{(19)}^{3, 0} )</td>
<td>0.06</td>
<td>0.9</td>
<td>0.01</td>
<td>0.002</td>
</tr>
</tbody>
</table>

A comparison is shown in Figure 7.

From the experiments, we make the following observations:

- Our algorithm shows good performance compared to GMCCE, MCCE and SPSA. Our algorithm also exhibits good global convergence behaviour in all the test cases. These illustrations corroborate the findings of Theorem 6. Note that GMCCE performs better than MCCE in the experiments. The better rate of convergence of GMCCE when compared to MCCE is already shown in [10]. However, the slower rate of these two algorithms when compared to our algorithm is primarily attributed to the insufficient samples. Even though we have chosen reasonably large sample size, it seems insufficient to match the performance of our algorithm. The algorithm also exhibits robustness w.r.t. the initial distribution \( \theta_0 \). An initial distribution which weighs the solution space reasonably well, seems to be sufficient. The values of the parameters \( \lambda_k, c \) and \( \epsilon_1 \) are same for all test cases. This implies that these parameters require minimal tuning in most cases. However, as with any stochastic approximation algorithm, the choice of the learning rates \( \alpha_k, \beta_k \) is vital.

- We studied the sensitivity of the algorithm with regards to the quantile factor \( \rho \). See Figure 6. For \( \rho \in \{0.4, 0.3, 0.2, 0.1\} \), the convergence of the algorithm is fast. However for \( \rho \in \{0.5, 0.01, 0.001, 0.0001\} \), the convergence is very slow. Theoretically, the algorithm should converge for all values of \( \rho \). However, in most practical cases, choosing \( \rho \) in the range \([0.3, 0.1]\) is highly recommended. Similar observation about the Monte-Carlo CE is mentioned in [8], and this needs further investigation.

- The computational and storage requirements of the algorithm are minimal. See Figure 7. This is attributed to the streamlined and incremental nature of our algorithm. This attribute makes the algorithm suitable in settings where the computational and storage resources are scarce.
the global convergence property of our algorithm.

The objective function $G(\cdot)$ in (37) can be rearranged as

$$
G(\lambda) = \left[ a_{11}, a_{21}, a_{31}\right] \left[ 1.0, 2.0 * e^{\lambda}, -2.0 * e^{\lambda}\right]^T +
\left[ e^{\lambda}, -e^{\lambda}\right] \begin{bmatrix}
 a_{22} & a_{23} \\
 a_{32} & a_{33} \\
\end{bmatrix}
\begin{bmatrix}
 e^{\lambda} & -e^{\lambda} \end{bmatrix}^T,
$$

where $A = (a_{ij})_{1 \leq i, j \leq 3} \triangleq \mathbb{E}[h_k h_k^\top]$ and $h_k = [r_{k+1}, a(s_k) - a(s_{k+1}), b(s_k) - b(s_{k+1})]^T$.

The various parameter values we used are as follows:

<table>
<thead>
<tr>
<th>$\varphi(\cdot)$</th>
<th>$\alpha_k$</th>
<th>$\beta_k$</th>
<th>$\lambda_k$</th>
<th>$c$</th>
<th>$\epsilon_1$</th>
<th>$\rho$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\exp (rx), r = 10^{-6}$</td>
<td>$0.9$</td>
<td>$k_0 = 0.03$</td>
<td>$0.95$</td>
<td>$0.1$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The results of the experiment are shown in Figure 8. The $x$-axis is the iteration number $k$. The performance measure used is Mean Squared Error (MSE) given by:

$$
\text{MSE}(\lambda) = \frac{\sum_{i=1}^{[S]} (V(i) - V_\lambda(i))^2}{\mu(i),}
$$

where $\mu(\cdot) \in \mathbb{R}^{[S]}$ is the stationary distribution of the Markov chain, i.e. $\mu$ satisfies $\mu^\top P = \mu^\top$.

Figure 8: Nonlinear Function Approximation. The plot shows the trajectory of $\sqrt{\text{MSE}}$ generated by TD(0), GTD2 and our algorithm against the iteration number $k$. Our algorithm returns the true value function $V$ which follows from the observation that $\sqrt{\text{MSE}}$ converges to 0. TD(0) slowly diverges, while GTD2 converges to a sub-optimal solution.

5 Conclusion

In this paper, we provided a novel multi-timescale stochastic approximation algorithm for the problem of stochastic optimization, where only noisy versions of the objective function are available. The convergence and the global optimization property of our algorithm are proven using the ODE method. The algorithm shows good performance on noisy versions of benchmark global optimization problems and gives promising results when nonlinear function approximators are used in the setting of Reinforcement Learning.

References