Gradient Descent on Evolutionary Strategy as a Blackbox Optimisation Problem

Chaitanya Kharyal and Tanmay Kumar Sinha

Contents

1 Introduction **1**

2 Interpolating Gradients from Directional Derivatives **1**

3 Results **2**

3.1 Evaluation on Rastrigin function 2

3.2 Genetic Image reconstruction problem 3

3.3 OpenAI Problems 3

1 Introduction

We propose a general method to solve black box optimization problems by approximately computing the gradient for the cost function, using a few calls to the function. We then implement this technique to train evolutionary algorithms for certain problems.

2 Interpolating Gradients from Directional Derivatives

We know that, given a cost function $f : \mathbb{R}^n \to \mathbb{R}$, we can write the directional derivative for f along a direction η as

$$\nabla_\eta f(x) = \lim_{t\to 0} \frac{f(x + t\eta) - f(x)}{t}$$

Also, we know a standard fact from multivariate calculus:- $\nabla_\eta f(x) = \eta \cdot \nabla f(x)$, where $\nabla f(x)$ is the gradient of f at x. Using the finite difference approximation, we can calculate $\nabla_\eta f(x)$ using at most 2 function calls(if we know $f(x)$, this only takes one function call). Hence, we can find an approximation for the gradient by solving the least squares problem

$$v = \text{argmin}_x \|\eta^T x - \nabla_\eta f(x)\|^2$$

We can improve our approximation by considering more directions. Let us say we choose m directions, $\eta_1, \eta_2 \cdots \eta_m$. Then, we get m linear equations $\eta_i^T \nabla f(x) =$

$\nabla_{\eta_i} f(x)$, and we can get an approximation of the gradient by solving the following least squares problem:-

$$v = \text{argmin}_x \|Ax - b\|^2$$

, where $A \in \mathbb{R}^{m \times n}$ is a matrix with rows η_i and b is vector such that $b_i = \nabla_{\eta_i} f(x)$

The whole process requires at most $m + 1$ calls to the function f, and also involves solving a linear least squares problem of size $m \times n$. Typically, if the time taken to compute the function is high(let us say that we compute the function by running some simulation), then we take $m << n$(e.g. in the examples we run later, n might be almost 4000, whereas taking m around 20 gives an optimal rate of learning).

Note that we **do not need any information about the form of the function** f - we only need it as a black box. For example, f might be the output of a simulation, with the parameters being the variables over which we optimize. One use case for this is embedding our algorithm into evolutionary algorithms, so that instead of making random perturbations, we can make a bit more informed ones. We actually ran the algorithm on a few such OpenAI problems, and got fairly good results, which are detailed below.

Algorithm 1 Basic Evolutionary Black Box Optimization Algorithm

```
Require: Initial population of parameters $x_1, x_2, \cdots x_k$
1: while not converged do
2:     Sort population according to their scores(let sorted order be $[x_1, x_2, \cdots x_k]$)
3:     for $i = 1$ to $k/2$ do
4:         Pick $m$ random vectors $\eta_1 \cdots \eta_m \in \mathbb{R}^n$
5:         $A = [\eta_1 \quad \eta_2 \quad \cdots \quad \eta_m]^T$
6:         $b = [(f(x_i + t\eta_1) - f(x_i))/t \quad \cdots (f(x_i + t\eta_m) - f(x_i))/t]^T$
7:         $v = \text{argmin}_x \|Ax - b\|^2$                    ▷ Approximate gradient
8:         $x_{k/2+i} = x_i - \alpha v$                    ▷ Add new parameters to the populatoin
```

3 Results

3.1 Evaluation on Rastrigin function

For establishing the Proof of Concept of our algorithm, we first benchmarked it on Rastrigin function, f, which has a lot of local minima and maxima.

$$f(x) = An + \sum_{i=1}^{n} [x_i^2 - Acos(2\pi x_i)]$$

We performed both gradient descent and ascent on benchmarking function, and our algorithm outperformed the classical genetic algorithm by a large margin in both.

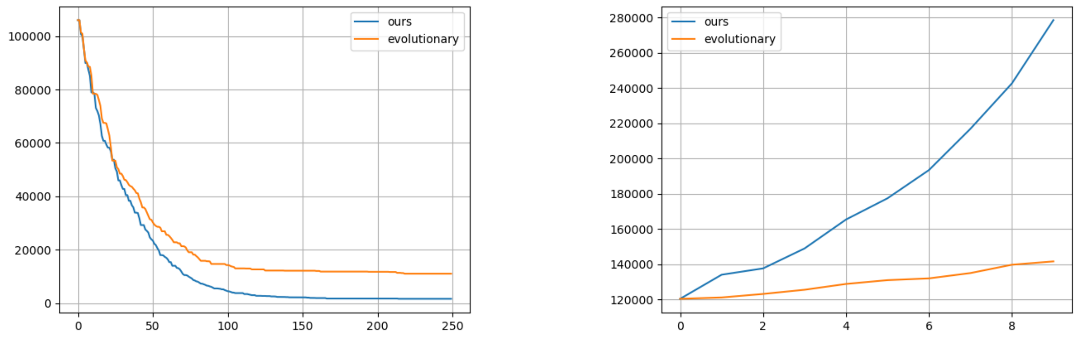

Figure 1: *results for solving the minimisation problem (left) and the maximisation problem (right)*

3.2 Genetic Image reconstruction problem

We also tested our algorithm against the classical genetic algorithm in an image reconstruction task. The problem was fairly difficult for a genetic algorithm to solve as the image had 16384 (128x128) parameters to solve for. For the blackbox loss function, we chose MSE loss which is relatively easy to descend on.

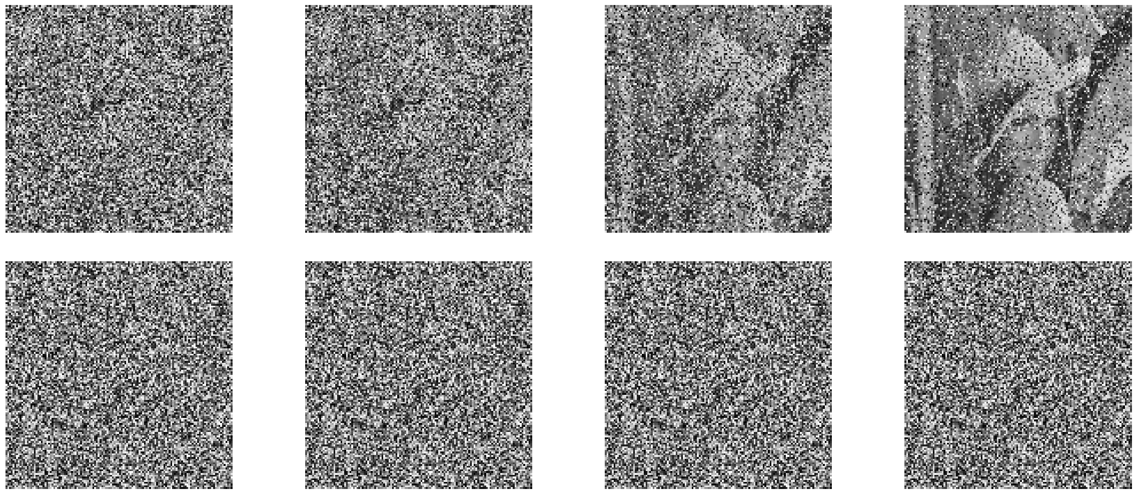

Figure 2: *Best images at 100, 200, 500, 1000 iterations for the proposed algorithm (top) vs the classical genetic algorithm (bottom)*

3.3 OpenAI Problems

We compared our algorithm against classical evolutionary strategies in a few OpenAI gym instances. Our algorithm performed much better than the classical algorithms. We ran it for the LunarLander problem and the CartPole problem.