Black Box Optimization using Genetic Algorithms

Varnit Mittal, Aditya Priyadarshi, Mohit Naik

September 2024

Overview

The goal of black box optimization is to locate the optimum of an unknown objective function $f(x)$, given a budget B of evaluations. When B is moderately large (>1000), we can use a genetic algorithm to find the global optimum, provided that the function is Lipschitz continous i.e. has bounded derivatives.

Problem Statement

We need to find the optimizer x^* of a black-box function f, i.e.

$$x^* = argmin(f(x))$$

given a budget B which specifies how many times you can evaluate the function at some input.

Algorithm

1. Maintain Population (Best Trials Set)

- Let S be the set of the best K_{best} trials observed so far.
- Let H be the history of all trials conducted.

At each iteration, the algorithm draws samples using one of three strategies.

2. Three Sampling Methods

1. Uniform Random Sampling:

- A point c is chosen uniformly from the feasible search space X. This is a basic random search strategy to ensure broad exploration.

  $$c \sim \mathcal{U}(X)$$

  where $\mathcal{U}$ denotes a uniform distribution over the feasible space X.

2. Ball Sampling (Genetic Mutation):

- A sphere is centered at the best point in S, denoted as x_{best}, with a radius r sampled from a power law distribution $P(r) \propto \frac{1}{r}$.

$$r \sim P(r) \propto \frac{1}{r}$$

- A new point c is sampled uniformly from the surface of this sphere.

$$c = x_{\text{best}} + r \cdot u$$

  where u is a uniformly sampled unit vector.

3. Linear Combination Sampling (Genetic Crossover):

- Two points x_a and x_b are randomly selected from the history H, with higher probability given to the best points from S.

- A new point c is generated as a linear combination of x_a and x_b, where the weight α is sampled from a normal distribution $\mathcal{N}(\mu, \sigma^2)$.

$$\alpha \sim \mathcal{N}(\mu, \sigma^2)$$

  The new point is:

$$c = \alpha x_a + (1 - \alpha) x_b$$

  This allows exploration along the line segment connecting x_a and x_b, and potentially beyond if α lies outside the range $[0, 1]$.

3. Update Best Trials Set

After generating a new candidate c, the algorithm evaluates the objective function $f(c)$. If c improves upon the worst point in S, it is added to S, and the worst point is removed.

Mathematical Explanation of Convergence

The convergence of the algorithm relies on shrinking the volume of the sub-level set $L(S_t)$ of the current best trials S_t. Specifically:

- Let $L(S_t) = \{x \mid f(x) \leq \max_{x \in S_t} f(x)\}$, representing the set of points with objective values no worse than the current maximum in S_t.

- Let $\mu(S_t)$ represent the volume of $L(S_t)$.

The convergence condition is:

$$\Pr\left[\mu(S_{t+1}) \leq (1-\eta)\mu(S_t)\right] \geq \delta$$

where η and δ are positive constants. This condition ensures that with high probability, the algorithm decreases the volume of the sub-level set over iterations, eventually converging to an approximate global optimum.

Intuitively, at every iteration of the algorithm, either S_t contains the approximate global optimum or the volume of S decreases. The function $f(x)$ being Lipschitz continuous ensures that there is a bounded rate of change. This condition prevents the function from having steep gradients, which helps in establishing stable regions of attraction around the optima. If $f(x)$ is Lipschitz continuous, the union of the regions of attraction around the global optima cannot have arbitrarily small volume. Random sampling guarantees that we will eventually point within that region of attraction

Properties of the Algorithm

- **Affine Invariance:** The algorithm is robust to affine transformations of the search space because linear combination sampling (crossover) is unaffected by such transformations. This ensures that the algorithm behaves similarly even if the feasible space is stretched or rotated.
- **Monotonic Transformations:** The algorithm depends only on the rank of the objective values rather than their absolute values. Therefore, it is robust to monotonic transformations of the objective function.

Pseudocode

- **Initialize:** Set history $H = \emptyset$, and best trials $S = \emptyset$.
- **For each iteration t:**
  1. With probability p_l, perform **Linear Combination Sampling**:
     - Sample $\alpha \sim \mathcal{N}(\mu, \sigma^2)$
     - Choose two points $x_a, x_b \in H$
     - Compute $c = \alpha x_a + (1-\alpha)x_b$
  2. With probability p_b, perform **Ball Sampling**:
     - Sample $r \sim P(r) \propto \frac{1}{r}$
     - Choose a point uniformly from the surface of a sphere around x_{best}
  3. Otherwise, perform **Uniform Random Sampling**:
     - Sample $c \sim \mathcal{U}(X)$

4. If $f(c) < \max(S)$, update S with c, replacing the worst point.

- **Return the best point found so far, after the budget is exhausted or convergence is achieved.**

More complex versions of the algorithm can be formulated to include noise in the objective function, and a set of constraints, which themselves can also be black-box.