Implementation

Caden Hewlett

2024-09-19

Introduction

Particle Swarm Optimization (PSO) is an evolutionary algorithm (EA) inspired by the schooling of fish or the flocking of birds. The algorithm optimizes an objective function by iteratively improving a candidate solution with respect to a given measure or quantity. It is a gradient-free technique, meaning that the objective function does not need to be differentiable.

The algorithm is initialized with a population of N candidate solutions called "particles." We henceforth denote the population of particles as $\mathfrak{X}_t = \left\{\mathbf{X}_t^{(1)}, \mathbf{X}_t^{(2)}, \dots, \mathbf{X}_t^{(N)}\right\}$, where $t \in \mathbb{N}$ is the iteration of the algorithm. Each individual particle $\mathbf{X}_t^{(i)}$ has a position in the search space, for $i \in [1, N]$. For example, if the objective function f maps $\mathbb{R}^d \mapsto \mathbb{R}$, then each particle $\mathbf{X}_t^{(i)}$ is a d-dimensional vector of positions in the search space, denoted $\mathbf{X}_t^{(i)} = \langle x_1^{(i)}, \dots, x_d^{(i)} \rangle_t$. In addition to positions, each particle has a vector of velocities $\mathbf{V}_t^{(i)}$ in d-dimensional space that evolves with each iteration. It should be noted that the objective function f is evaluated on the positions $f(\mathbf{X}_t^{(i)})$, and the change in positions between generations $\mathbf{X}_t^{(i)} \to \mathbf{X}_{t+1}^{(i)}$ is moderated by the velocity vector $\mathbf{V}_t^{(i)} = \langle v_1^{(i)}, \dots, v_d^{(i)} \rangle_t$. Each particle has its own position and velocity. The velocity $\mathbf{V}_t^{(i)}$ evolves between generations $\mathbf{V}_t^{(i)} \to \mathbf{V}_{t+1}^{(i)}$ by the *velocity update equation.* The position of a vector between generations is determined by adding the updated velocity to the current position, i.e.

$$\mathbf{X}_{t+1}^{(i)} = \mathbf{X}_t^{(i)} + \mathbf{V}_{t+1}^{(i)}, \text{ for } i \in [1, N],\ t \in \mathbb{Z}^+$$

Where $\mathbf{X}_t^{(i)}$ is the current position of particle i in generation t, $\mathbf{X}_{t+1}^{(i)}$ is the updated position at iteration $t+1$, and $\mathbf{V}_{t+1}^{(i)}$ is the updated velocity that controls the change in position.

Declaration of the Population

In order implement the algorithm, we must have some population $\mathfrak{X}_0$ at $t = 0$. In this work, we initialize the population via a chaotic sequence. Specifically we use a logistic map.

The Logistic Map

The Logistic Map depends on the parameter r, which controls the behaviour of the system and ranges between 0 and 4.

For $r \in [0, 1)$ the population will eventually die out, i.e. $x_n \to 0$. For $\mu \in [1, 3)$ the system stabilizes to a fixed point dependent on μ, and for $\mu \in [3, \approx 3.57]$ the system exhibits period-doubling bifuractions between two points. Finally, for $r \in [3.57, 4]$ the system is chaotic and the values of x_n are highly dependent on the initial condition x_0.

The general evolution of a logistic map is given by:

$$x_{n+1} = \mu \cdot x_n \cdot (1 - x_n), \text{ where } \mu \in [0, 4] \text{ and } x_0 \in \mathbb{R}$$

Below is a bifurcation diagram of the logistic map with $x_0 = 0.35$ across μ values. We see that the system becomes increasingly chaotic past $\mu \approx 3.57$.

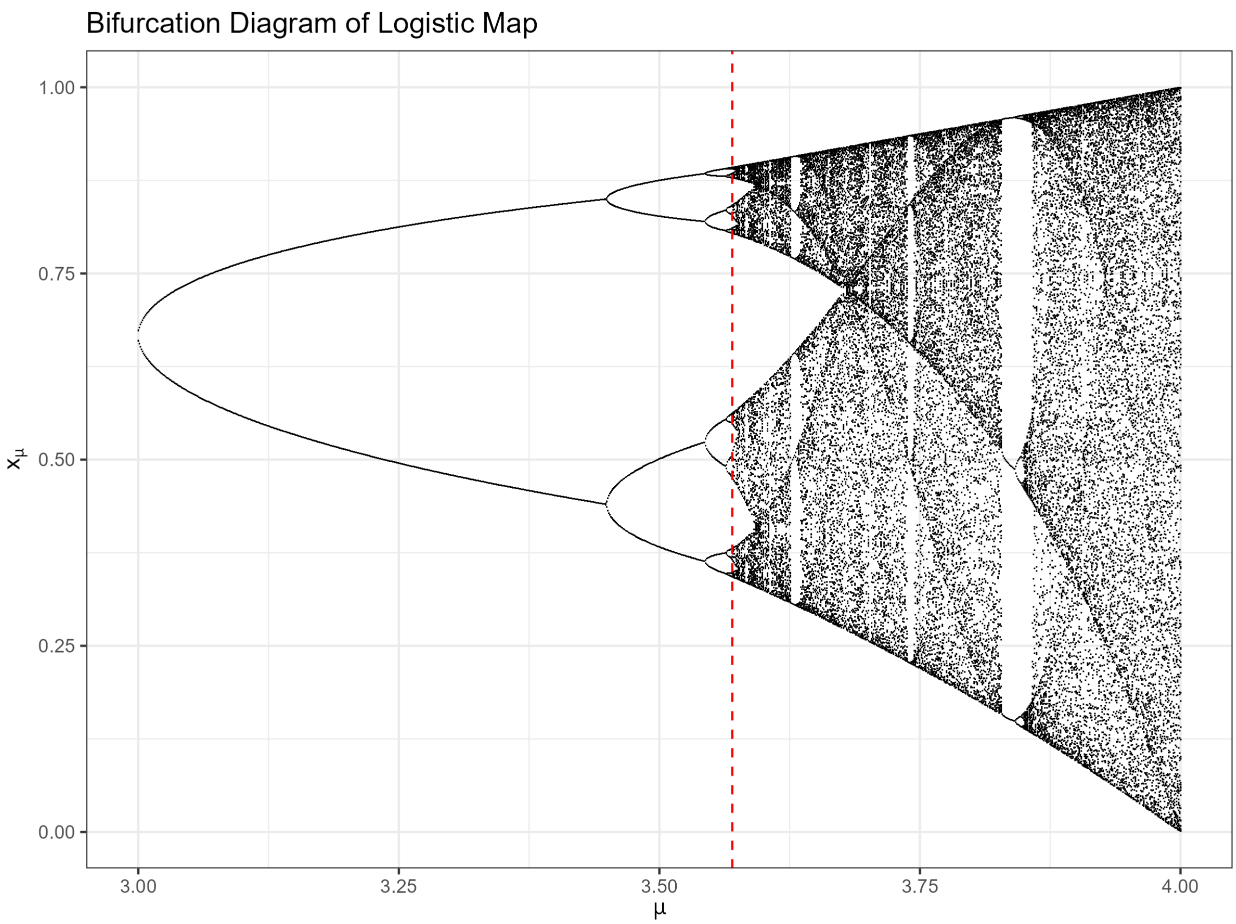

Figure 1: bifurcation