Optimization Techniques Mini Project

Optimizing Deep Neural Network Weights using Particle Swarm Optimization

Viraj Vora, 231070078, TY BTECH CE
Kevin Shah, 231070060, TY BTECH CE

1. Problem Statement: Non-Gradient Optimization for DFNNs

The Overarching Research Problem

The fundamental challenge in training Deep Feedforward Neural Networks (DFNNs) is minimizing a non-convex loss function across millions of weights. While gradient-based methods (like Backpropagation with Adam or SGD) are standard, they often suffer from convergence to poor local minima or vanishing/exploding gradient issues.

The research paper "Optimizing Neural Network Weights Using Nature-Inspired Algorithms" proposes and evaluates **Nature-Inspired Meta-Heuristics**—specifically Particle Swarm Optimization (PSO), Mother Tree Optimization (MTO), and MTOCL—as robust, non-gradient alternatives. The goal is to provide a viable, global search method that can efficiently train DFNNs and achieve superior performance, particularly on complex classification tasks such as the Wisconsin Breast Cancer Dataset (WBCD).

The Specific Implementation Problem

A major practical hurdle for meta-heuristics is **computational speed**. Standard PSO, which evaluates the entire population's fitness (the full forward pass) in every iteration, is often substantially slower than optimized gradient descent (GD).

The specific problem addressed by the provided Jupyter Notebook implementation is:

How can the PSO algorithm be aggressively optimized to be competitive with, or even faster than, modern GD methods while maintaining sufficient classification accuracy, thereby demonstrating its practical viability for neural network weight optimization?

The implementation sacrifices the expensive, deep global exploration of a full PSO run in favor of a "Fast PSO" model designed for maximum efficiency, focusing squarely on improving the speed-to-accuracy trade-off.

2. Approach: "Fast PSO" Implementation and Trade-off Analysis

The adopted approach utilizes a meta-heuristic framework where the DFNN's entire weight and bias structure is flattened into a high-dimensional vector, representing the **position of a single PSO particle**. The **fitness function** is the Binary Cross-Entropy Loss, which the swarm is tasked to minimize via iterative updates based on inertia, cognitive (personal best), and social (global best) components.

To overcome the performance gap with gradient descent, the notebook implements aggressive speed optimizations:

Key Speed Optimization Strategies

- **Vectorization with NumPy & Numba JIT:** The core PSO update logic (velocity, position, PBest, GBest) is fully vectorized using NumPy. Crucially, the activation functions (ReLU, sigmoid) are compiled using **Numba's Just-In-Time (JIT)** compiler. This eliminates slow Python loops and allows critical calculations to run efficiently, drastically speeding up the repeated forward passes.
- **Reduced Function Evaluations:** The implementation uses a small population size (10-15 particles) and a limited number of iterations (e.g., 60-80) with early stopping (15 iterations of no improvement). This strategy minimizes the total number of computationally expensive network forward passes required for convergence.
- **Simplified PSO and Efficient NN Model:** A standard constriction PSO with aggressive convergence parameters (e.g., $w=0.7298$) is employed, avoiding complex local search routines. The custom FastNeuralNetwork class uses an efficient, batch-based forward pass and is initialized using **He/Xavier methods** to start the search closer to optimal solutions, further reducing convergence time.

3. The most relevant reference papers we referred to:

https://arxiv.org/abs/2105.09983

4. The Dataset link:

https://archive.ics.uci.edu/dataset/17/breast+cancer+wisconsin+diagnostic

5. The results

The comparative analysis against the highly optimized Scikit-learn MLPClassifier (using the Adam solver) confirms the inherent trade-off:

Dataset	PSO Test Accuracy	GD Test Accuracy	PSO Speed (Relative to GD)
Small Synthetic	0.8500	0.7750	6.14x Slower
Wisconsin Cancer	0.9649	0.9825	1.64x Slower

Despite being consistently slower than the optimized GD baseline—which benefits from high-level C/C++ implementations—the Fast PSO approach demonstrated **competitive accuracy** (e.g., 96.49% on the WBCD). In the small synthetic case, PSO even achieved **7.50% higher accuracy**.