**FIREFLY SPARK CLASSIFICATION OPTIMIZATION**

Abstract: Classification is a problem at the forefront of computer science. However, models for classification remain extremely computationally expensive. Therefore, stochastic algorithms provide a more efficient manner of model training. Furthermore, the Apache Spark context provides parallel processing capabilities in order to further improve classifier efficiency through increasing concurrency.

**01. INTRODUCTION**

In line with the old adage “more is better,” computer scientists have been trying to develop parallel processes to speed up large-scale data processing for decades. In 2006 Apache released its first parallelization framework Hadoop, which enabled batch processing of large-scale datasets. The Hadoop package utilized the Map-Reduce framework of data flow models to efficiently and effectively solving and evaluating problems. The Hadoop Distributed File System (HDFS), which is a fault-tolerant large-scale file storage system for parallel computing, was a central component of this system. Spark, which incorporates the Hadoop Distributed File System (HDFS) into its architecture, was created by the UC Berkeley AMPLab. Spark’s Resilient Distributed Datasets (RDD), increased computational efficiency, real-time processing, and improved memory management, which resulted in a scalable, low-latency, general framework for at-scale computation. These developments in classification learning problems, specific individuals into specific groups. Algorithms for classification tasks in machine learning have existed since the 1940s, however, with the increasing computational capabilities of the 21st century, classification has become an increasingly important part of modern machine learning. Since the classification of NP-hard problems, scientists have tried to develop efficient deterministic algorithms. However, as these algorithms are currently mathematically intractable, scientists instead turn towards less accurate, but much faster stochastic algorithms. These processes test only a small proportion of the sample space instead of searching its entirety leading to their non-deterministic nature. One popular area of exploration is bio-inspired algorithms that are based on the observation of wild animals. These algorithms provide a more extendable solution compared to gradient descent functions.

**02 PROBLEM DESCRIPTION**

Swarm Intelligence (SI) has become increasingly important in stochastic search algorithms. These algorithms are inspired by wild species’ social behavior and movement patterns. The Firefly algorithm was authored by Xin-She Yang in his 2008 publication while working at Cambridge. The Firefly algorithm is a subset of bio-inspired algorithms based on the movement of fireflies. The Firefly algorithm randomly moves in the swarm toward the brightest or the fitted direction for a defined number of iterations. Apache Spark is a relatively new parallel computing network based on the idea of a master-worker architecture of nodes executing similar algorithmic steps on different segments of the data. This structure benefits nature-inspired algorithms as each individual particle in the algorithm can be processed independently by a worker node. Spark will then enable to maintain data consistency per node in every particle in each iteration of the Firefly algorithm to determine the optimal centroids for data evaluation. As the Firefly algorithm is stochastic, it requires fewer evaluations to find global maxima. Several modern stochastic algorithms exist for classification, however, applying Wolf’s no-free-lunch theorem to classification averaged across all datasets all stochastic classification algorithms retain the same accuracy. Thereby, the implementation of any one algorithm will be equally applicable to any other. The use of a simplistic centroid-based vector Euclidean algorithm is thereby justified to predict target parameters and demonstrate the power of bio-inspired Spark integrated algorithms.

**03 RELATED RESEARCH**

There is significant promise in using the Spark framework to parallelize many computationally intensive computer science problems. In fact, it has been proven to limit the curse of dimensionality problem that plagues popular swarm optimization algorithms. Many Swarm Intelligence algorithms have been used in classification, the simplest of these is the Particle Swarm (PSO) algorithm. PSO is a direct extension the Firefly algorithm that works on both migration and hunting patterns. PSO is the most commonly used stochastic algorithm for classification, however, PSO suffers from local convergence where the particles converge too quickly to local minima instead of finding the global extremum. The Firefly algorithm converges based on the total fitness of surrounding individuals and a complex movement equation, which creates a less direct convergence pattern. The Firefly algorithm mitigates PSO’s convergence problem through path and velocity vector optimization that accounts for the total fitness of the space related to distance instead of just the current global best. One popular optimization of the PSO algorithm is the use of more points in movement computation called local and global clustering to create a less deterministic movement of each particle. In fact, the higher topology of integration between particles in computing the stochastic numerical optimization algorithms the higher the probability of reducing a true maxima. The Firefly algorithm takes this principle to the extreme by exponentially increasing the interactions between particles and can be optimized through the parallel processing ability of Apache Spark. The Firefly algorithm has been well-documented in various NP-hard scenarios prompting speed-up observations for PSO and other machine learning models.

**04 METHODOLOGY**

The implementation of the Firefly algorithm shall rely on three central parts: fitness, brightness, and movement equations. These equations will serve as the mathematical core of the algorithm while Spark will enable these equations to process asynchronously in Java. There are alternative implementations to each of these equations that exist as optimizations to the Firefly algorithm. However, these are the most common iterations referenced in the original 2008 publication. The fitness function for the evaluation of the fitness of a particle will be based on the Euclidean Distance function, the fitness of a particle shall be the sum of the distances between it and every other point p in the dataset with the same classification target. The brightness at particle i of the particle y based on the number of data records (n) is derived from the inverse square law. The movement between particles and y given a random constant fc. While these equations form the mathematical foundation for the algorithm its scalability is found in loop concurrency when evaluating particle fitness and movement that enable Spark to scale at execution.

**05 RESULTS**

The implemented algorithm was run in Java 8 on the North Dakota State University Spark Cluster. The Spark Cluster has 6 worker nodes, each with 1024MB of memory and 8 cores allowing it to physically run 6 executives and virtually manage more. The data was run on the publicly available binary EEG-eye state data records, measuring the eyelid’s response to closed-eye movement with 45 thousand data records. The output, measuring the outcome of poker hands, with 1.2 million records to demonstrate the difference in speed-up based on the number of records and differences between binary and multi-classification targets. Each dataset was run with 1, 2, 4, and 64 nodes in order to capture Spark’s virtualization tendency. Each classification target had 100 particles initialized for it and the algorithm ran for 150 iterations. The EEG dataset has less proportionate speed up as the number of nodes increased while maintaining higher accuracy. The shifts in accuracy are caused by the random nature of the swarm initialization and as the Firefly algorithm is stochastic, deterministic results are not guaranteed resulting in changing accuracies. The speed-up values increased more proportionally with the number of nodes in the poker dataset demonstrating the difference between the binary targets of the EEG dataset and the ten multi-targets of the poker dataset as well as the eighty-factor difference in their size. In addition, the drop-off in speed-up after 6 nodes is due to the physical limitations of the cluster itself, which only has 6 worker nodes. Past this point, Spark is virtually no nodes meaning that instead of adding physical computation Spark is jobs scheduling and switching on the same executor to simulate extra executors. Therefore, the speed-up values do not continue to linearly increase. In addition, the generally low accuracy of the EEG dataset could mean a lack of particles as both have 100 particles per classification target, and as the dataset multiplied the algorithm failed to exploit the larger dataset for appropriate cluster groupings.