Swarm-based Algorithms for Neural Network Training

Reginald McLean

Submitted in partial fulfilment
of the requirements for the degree of

Master of Science

Department of Computer Science
Brock University
St. Catharines, Ontario

Abstract

The main focus of this thesis is to compare the ability of various swarm intelligence algorithms when applied to the training of artificial neural networks. In order to compare the performance of the selected swarm intelligence algorithms both classification and regression datasets were chosen from the UCI Machine Learning repository. Swarm intelligence algorithms are compared in terms of training loss, training accuracy, testing loss, testing accuracy, hidden unit saturation, and overfitting.

Our observations showed that Particle Swarm Optimization (PSO) was the best performing algorithm in terms of Training loss and Training accuracy. However, it was also found that the performance of PSO dropped considerably when examining the testing loss and testing accuracy results. For the classification problems, it was found that firefly algorithm, ant colony optimization, and fish school search outperformed PSO for testing loss and testing accuracy. It was also observed that ant colony optimization was the algorithm that performed the best in terms of hidden unit saturation.

Acknowledgements

To my parents for their unconditional love and support throughout my education and my future endeavours. To my sister for always being willing to read about many Computer Science topics that she does not fully comprehend. To my partner for her constant love and support through many long stressful nights. Lastly, to Prof. Ombuki-Berman and Prof. Engelbrecht for their support, guidance, and extensive expertise in the field that allowed me to complete this work.

Contents

1 Introduction **1**

2 Background Information **6**

2.1 Swarm Intelligence Algorithms 6

2.1.1 Particle Swarm Optimization 6

2.1.2 Ant Colony Optimization 8

2.1.3 Artificial Bee Colony Optimization 9

2.1.4 Bacterial Foraging Optimization Algorithm 11

2.1.5 Bat Algorithm 12

2.1.6 Firefly Algorithm 14

2.1.7 Fish School Search Algorithm 15

2.2 Artificial Neural Network 18

2.3 Activation Functions 20

2.3.1 Sigmoid Function 20

2.3.2 Hyperbolic Tangent Function 21

2.3.3 Lecun's Hyperbolic Tangent 22

2.3.4 Elliot 22

2.4 Saturation 23

2.4.1 Measure of Hidden Unit Saturation 24

2.5 Overfitting 25

2.5.1 Indication of Overfitting 26

3 Swarm Intelligence Algorithms Previous Work **27**

3.1 Particle Swarm Optimization 27

3.2 Artificial Bee Colony Optimization 29

3.3 Ant Colony Optimization 30

3.4 Bacterial Foraging Algorithm 30

3.5 Bat Algorithm 31

3.6 Firefly Algorithm . . . 31
3.7 Problem Definition . . . 32

4 Implementation **33**
4.1 Particle Swarm Optimization . . . 33
4.2 Artificial Bee Colony . . . 34
4.3 Ant Colony Optimization . . . 35
4.4 Bat Algorithm . . . 35
4.5 Bacterial Foraging Optimization . . . 35
4.6 Fish School Search . . . 35
4.7 Firefly Algorithm . . . 36

5 Experimental Setup **39**
5.1 Neural Network Architectures . . . 39
5.1.1 Defining Number of Hidden Neurons . . . 41
5.2 Parameter Tuning for each experiment . . . 42
5.3 Experimental Parameters . . . 42

6 Results **44**
6.1 Regression Results . . . 45
6.2 Classification Results . . . 45
6.3 Overall Results . . . 46
6.4 Saturation Results . . . 52
6.5 Overfitting Results . . . 55

7 Conclusion **58**
7.1 Performance of SI Algorithms . . . 58
7.2 Future Work . . . 59

Bibliography **66**

Appendices **67**

A Additional Experimental Analysis **67**
A.1 Parameters . . . 67
A.2 Average and Standard Deviation of Results . . . 95

List of Tables

5.1 Neural Network Architectures for each Dataset 40
5.2 PSO with Tanh Activation function parameters 43

6.1 Testing Loss Regression Datasets 46
6.2 Testing Loss Classification Datasets 47
6.3 Testing Accuracy Classification Datasets 48
6.4 Testing Loss Overall Results 49
6.5 Testing Loss Small Dataset Ranks 50
6.6 Testing Loss Medium Dataset Ranks 50
6.7 Testing Loss Large Dataset Ranks 50
6.8 Testing Accuracy Small Dataset Ranks 51
6.9 Testing Accuracy Medium Dataset Ranks 51
6.10 Testing Accuracy Large Dataset Ranks 51
6.11 Saturation Rankings for Regression Datasets 52
6.12 Saturation Results for Classification Datasets 53
6.13 Saturation Results 54
6.14 ACO Saturation Results Compared to Training and Testing Loss . . . 55
6.15 Overfitting Results 57

A.1 PSO with Sigmoid Activation function 68
A.2 PSO with Elliot Activation function 69
A.3 PSO with Lecun Activation function 70
A.4 ABC with Tanh Activation function 71
A.5 ABC with Sigmoid Activation function 72
A.6 ABC with Elliot Activation function 73
A.7 ABC with Lecun Activation function 74
A.8 ACO with Tanh Activation function parameters 75
A.9 ACO with Sigmoid Activation function Parameters 76
A.10 ACO with Elliot Activation function parameters 77

A.11 ACO with Lecun Activation function parameters 78
A.12 BA with Tanh Activation function parameters 79
A.13 BA with Sigmoid Activation function parameters 80
A.14 BA with Elliot Activation function parameters 81
A.15 BA with Lecun Activation function parameters 82
A.16 BFA with Tanh Activation function parameters 83
A.17 BFA with Sigmoid Activation function parameters 84
A.18 BFA with Elliot Activation function parameters 85
A.19 BFA with Lecun Activation function parameters 86
A.20 FSS with Tanh Activation function parameters 87
A.21 FSS with Sigmoid Activation function parameters 88
A.22 FSS with Elliot Activation function parameters 89
A.23 FSS with Lecun Activation function parameters 90
A.24 Firefly with Tanh Activation function parameters 91
A.25 Firefly with Sigmoid Activation function parameters 92
A.26 Firefly with Elliot Activation function parameters 93
A.27 Firefly with Lecun Activation function parameters 94
A.28 Training Loss Results 95
A.29 Training Loss Results 96
A.30 Training Loss Results 97
A.31 Training Loss Results 98
A.32 Training Accuracy Results 99
A.33 Training Accuracy Results 99
A.34 Training Accuracy Results 100
A.35 Training Accuracy Results 100
A.36 Testing Loss Results 101
A.37 Testing Loss Results 102
A.38 Testing Loss Results 103
A.39 Testing Loss Results 104
A.40 Testing Accuracy Results 105
A.41 Testing Accuracy Results 105
A.42 Testing Accuracy Results 106
A.43 Testing Accuracy Results 106
A.44 Saturation Results 107
A.45 Saturation Results 108
A.46 Saturation Results 109

A.47 Saturation Results 110
A.48 Overfitting Indicator Values 111
A.49 Overfitting Indicator Results 112
A.50 Overfitting Indicator Results 113
A.51 Overfitting Indicator Results 114
A.52 Training Loss Regression Datasets 114
A.53 Training Loss Classification Datasets 115
A.54 Training Accuracy Classification Datasets 116
A.55 Overall Training Loss Results 117

List of Figures

2.1 A Simple Neuron 18
2.2 An Artificial Neural Network 20
2.3 The Sigmoid function. 21
2.4 The Hyperbolic Tangent function. 21
2.5 The Lecun Hyperbolic Tangent function. 22
2.6 The Elliot function. 23

Chapter 1

Introduction

This thesis is a comparative study of several swarm-based algorithms and their abilities to train artificial neural networks (ANN). In this thesis, the accuracy, loss, and saturation of the ANN generated by each algorithm is compared to the ANN produced by each swarm Intelligence algorithm.

An ANN is a versatile machine learning model that is able to succeed in a wide range of tasks including but not limited to: image recognition, speech recognition, or recommendation engines. They consist of a set of nodes that are interconnected to each other with a weighted connection. The focus of this work will be on ANNs of only three layers: an input layer, a hidden layer, and an output layer. The input layer receives data into the network, the hidden layer applies a nonlinear function to the inputs, and the output layer attempts to generate a relevant answer to the task at hand.

ANNs are generally trained using the backpropagation learning algorithm. This algorithm uses how incorrect the ANN is and attempts to adjust the weight between each node appropriately. This type of learning ideally achieves a level of generalization that is acceptable where the ANN can accurately output values that are correct for unseen data.

Alternatively, ANNs can be trained using swarm-based algorithms, such as par-

ticle swarm optimization (PSO). The PSO algorithm iteratively solves a problem by updating a population of solutions to a problem using simple mathematical formulas applied to a particles position and velocity [21]. These updates are also influenced by the particles own best found position, as well as the best found position in the population overall.

Past research has shown that the application of PSO to training ANNs has shown some deficiencies. It has been found that PSO suffers from hidden unit saturation, while also having a tendency to generate ANNS that are overfit [38]. The main contribution of this thesis is to determine if other swarm-based algorithms saturate like PSO does, while also comparing the swarm-based algorithms in terms of testing loss & accuracy, and overfitting behaviour. These other algorithms include: ant colony optimization [31], artificial bee colony optimization [18], bacterial foraging optimization [27], bat algorithm [46], firefly algorithm [45], and fish school search optimization [6].

The ant colony optimization (ACO) algorithm is based on the innate ability of ants to find the shortest path to a food source based on the amount of pheromone that other ants have deposited on a path [31]. The more of the pheromone that is deposited on a path, the more likely that an ant will travel down that path. The ACO algorithm originally was designed to optimize discrete optimization problems and the algorithm has been adapted to optimize ANN training [31].

The artificial bee colony (ABC) algorithm is inspired by the intelligent behaviour of a honey bee swarm. In the ABC algorithm, the population is divided into three different parts: employed bees, onlooker bees, and scout bees [18]. Employed bees go to their food source and then return to the hive and dance on this area [18]. If an employed bees food source becomes abandoned, this bee will become a scout bee and search for a new food source. Onlooker bees watch the dances of employed bees and choose a food source based on the dance [18].

The bacterial foraging optimization (BFO) algorithm is inspired by the group foraging behaviour of E. coli and M.xanthus bacteria [27]. Specifically from these bacteria, the behaviour is inspired by the chemotaxis that perceives the chemical gradients in the environment and moves the bacteria towards or away from the source [27].

The bat algorithm is influenced by the echolocation techniques of microbats [46]. Each bat in the population flies randomly and modifies its echolocation impulses when it finds prey. The position of bats is updated in an iterative manner similar to PSO where a velocity and position update are influenced by previously found best solutions [46].

The firefly algorithm is inspired by the flashing behaviour of fireflies [45]. Based on this principle, fireflies are attracted to any other individual in the population. Fireflies move towards each other based on the brightness that they are currently exhibiting with the lower brightness firefly being attracted by the brighter one [45]. The brightness is related to the function that is being optimized.

The fish school search algorithm is inspired by the collective behaviour of a school of fish [6]. The school swims towards the better solutions in order to feed and gain weight. These feeding and weight gain operators influence the direction of the school which moves the fish into the areas where better solutions may be found [6].

In order to compare the abilities of these algorithms, 23 datasets were chosen from the UCI machine learning repository. There are 7 regression datasets and 16 classification datasets. The regression datasets are Forest Fires, Geographical Original of Music, Residential Building, Facebook Metrics, Auto, Computer Hardware, and Servo. The classification datasets are: Iris, Glass, Zoo, Wine, Parkinsons, Soybean, SCADI, Ionosphere, Musk V2, Breast Cancer, Connectionist, Thyroid, Seizures, Land Cover, Spambase, and MNIST.

This thesis focuses on a few measures specifically related to the ANNs ability to

generalize for both classification and regression problems. The algorithms will search for a set of connection weights by using 80% of the dataset as a training set. Once the training has completed, the remaining 20% of the dataset is used as the testing set.

The experiments in this thesis use every combination of algorithm, activation function, and dataset to compare how the other swarm-based algorithms perform relative to PSO. The measures that are collected for each algorithm include: training accuracy, training loss, testing accuracy, testing loss, saturation of hidden nodes, and a measure of overfitting.

It was found that particle swarm optimization was the best algorithm when examining the training accuracy and training loss, but performance suffered when applied to the testing data. The overfitting indicator was used at this point to discover that particle swarm optimization was likely to overfit, which follows the results found when examining training versus testing results.

When using classification datasets, ant colony optimization, firefly algorithm, and fish school search all surpassed particle swarm optimization. Particle swarm optimization was the best performing algorithm for the regression datasets.

Ant colony optimization was found to be the algorithm that saturated the least, while also examining the relationship between the performance of Ant Colony Optimization and how much or how little Ant Colony Optimization saturated in those datasets. There is a relationship to be examined which follows the previous literature.

This thesis is organized as follows: Chapter 2 includes an introduction to artificial neural networks. Chapter 2 also includes a detailed exaplanation of particle swarm optimization, ant colony optimization, artificial bee colony optimization, bacterial foraging optimization, bat algorithm, firefly algorithm, and fish school search. Lastly, Chapter 2 describes saturation and how this thesis will measure it, as well as overfitting and how an overfitting indicator will be used in this thesis. Chapter 3 of this

thesis describes the literature in which swarm intelligence algorithms have been applied previously to training artificial neural networks. Chapter 4 describes how each algorithm was implemented, Chapter 5 contains the parameters used for the artificial neural network architectures, as well as the parameters for each algorithm. Chapter 6 contains the rankings results for each of: loss, accuracy, saturation, and overfitting. Lastly Chapter 7 summarizes the results found and proposes some future work.

Chapter 2

Background Information

2.1 Swarm Intelligence Algorithms

Swarm intelligence (SI) algorithms use a system of agents that interact with their environment and other agents to guide the search to solve a problem []. This problem can be either a continuous or discrete optimization problem. These algorithms have the following properties:

- Composed of many individuals.
- Either identical individuals, or very few subclasses.
- Interactions between individuals follow simple rules that exploit only information exchanged within the population or from interacting with the environment.
- This leads to the overall group behaviour of self-organization.

This thesis focuses on algorithms that are categorized as SI algorithms.

2.1.1 Particle Swarm Optimization

Particle swarm optimization (PSO) is a metaheuristic introduced by Kennedy and Eberhart [21]. The PSO algorithm was inspired by the flocking behaviour of birds.

PSO maintains a population of solutions that are iteratively updated to move in the search space. Mathematical formulae are applied to every particle's velocity and position. The velocity of a particle controls how fast the particle travels through the search space while the position of a particle represents the solution to the problem being explored. The calculations at each iteration are based on two positions that were previously found; the personal best location and the best position found in a neighbourhood. The particles are ranked based on the quality of the solution that they produce. In minimization problems, the best particle positions are the ones that produce the smallest fitness based on the function applied. The opposite is true for a maximization problem. The neighbourhood is a representation of how information is shared between particles; in a fully connected neighbourhood every particle has access to all of the information available, while other network topologies limit the amount of information available to each particle.

The velocity of a particle is calculated at each iteration using the following equation:

$$v_i(t+1) = \omega v_i(t) + c_1 r_1(y(t) - x_i(t)) + c_2 r_2(\hat{y}(t) - x_i(t)) \tag{2.1}$$

where t is the current iteration, $x_i(t)$ is the current position of the particle at dimension i, $v_i(t)$ is the current velocity of the particle at dimension i, $y_i(t)$ is the personal best position of the current particle at dimension i, $\hat{y}_i(t)$ is the global best position at dimension i, ω is the inertial term, which applies a portion of the previous velocity to the next velocity, C_1 is the cognitive component, which influences the effect of the personal best found position, C_2 is the social component, which influences the effect of the global best found position, and R_1 and R_2 are random values in the range [0, 1]. The parameters ω, C_1, and C_2 have a major effect on the performance of the PSO algorithm. Typically the inertial term is in the range [0, 1], and the cognitive and social terms are in the range [0, 2]. The position of a particle is then updated

using the following formula:

$$x_i(t+1) = x_i(t) + v_i(t+1) \tag{2.2}$$

At every iteration, the updated velocity of a particle is added to the current position to generate the position of the particle in the subsequent iteration.

2.1.2 Ant Colony Optimization

The ant colony optimization (ACO) algorithm was introduced in [10] with aims to search for an optimal path in a graph. The ACO method is based on the ability of ants to search for a path between their colony and a food source. Initially, the ants will randomly choose paths to the food source. As the ants travel, they deposit pheromone along their path. This deposit leaves information for the next set of ants to travel to the food source using the same path.

The ACO algorithm has been extended to optimize continuous valued problems, such as training an ANN. The initialization phase of ACO uses the number of inputs, outputs, and number of hidden nodes in the ANN to determine the length of the path to generate. Once the size of the path is found, each connection weight in the ANN is divided into d discrete points generated from a normal distribution. Additionally, each connection weight is also assigned an initial pheromone value using the formula:

$$\tau_{ijh} \leftarrow 1/(n_i + n_h + n_o) \tag{2.3}$$

where n_i is the number of inputs, n_h is the number of hidden nodes, and n_o is the number of output nodes.

The ants in ACO each generate a solution to the problem to be optimized. At each iteration an ant probabilistically chooses a path to travel on based on pheromone information at that point. This probabilistic choice is generated through the following

formula:

$$\rho_{ijh} = \frac{\tau_{ijh}}{\sum_{k=1}^{d} \tau_{ijk}} \tag{2.4}$$

where ρ_{ijh} is the probability of selection path component A_{ijh} for W_{ij}, d represents the number of discrete points, and τ_{ijh} represents the existing pheromone trail associated with A_{ijh}. Once each ant has generated a complete solution the pheromone trails are updated. Only the best ant updates the pheromone of the trails, using the formula:

$$\tau_{ijh} \leftarrow \tau_{ijh} + \Delta_{\tau ijh}^{best}, \forall a_{ijh} \in T^{best} \tag{2.5}$$

where T^{best} is the best found combination of points, and $\Delta_{\tau ijh}^{best}$ is the amount of pheromone to be deposited which is defined as:

$$\Delta_{\tau ijh}^{best} = \frac{1}{E^{best}} \tag{2.6}$$

where E^{best} is the fitness of the solution generated by the best ant. Here, the better the quality of the solution, the lower E^{best}, the more pheromone that is deposited. The pheromone is then updated using pheromone evaporation:

$$\tau_{ijh} \leftarrow (1 - \rho)\tau_{ijh}, \forall a_{ijh} \tag{2.7}$$

where ρ is a constant in the range [0, 1]. This evaporation helps eliminate poor decisions made in previous iterations.

2.1.3 Artificial Bee Colony Optimization

The artificial bee colony (ABC) algorithm was introduced in [18]. The ABC algorithm is based on the collective behaviour of a bee colony in the search for food sources. ABC uses three different types of bees: employed bees, onlooker bees, and scout bees. The employed are located at a food source. These bees represent a solution to the

problem. A food source is generated using the formula:

$$x_{mi} = l_i + rand(0,1) * (u_i - l_i) \tag{2.8}$$

where l_i and u_i are the lower and upper bounds for the position i, respectively.

The employed bees will search locally around their food source for other potential food sources. The number of employed bees is equal to the number of food sources found around the hive. Employed bees will determine a neighbouring food source using the following formula:

$$v_{mi} = x_{mi} + \phi_{mi}(x_{mi} - x_{ki}) \tag{2.9}$$

where x_k is a randomly selected food source, i is randomly chosen parameter index, and ϕ_{mi} is a random number within the bounds (l_i, u_i).

The onlooker bees will search around the hive for other food sources, based on the probability generated by the quality of the food sources that the employed bees are currently examining. Once a food source is probabilistically chosen, the onlooker bee then searches near that food source for a new food source. The probabilistic calculation is performed using the following formula:

$$p_m = \frac{fit_m(\vec{x_m})}{\sum_{m=1}^{SN} fit_m(\vec{x}_m)} \tag{2.10}$$

Once a food source is chosen, a new food source is created using Equation 2.9. If the fitness value of the new position is better than the current position, the current position is abandoned and the new position is used.

The last bee type is the scout bee. This type of bee attempts to find new food sources for employed bees once the previous food source has reached a set number of trials for searching by employed and scout bees. The new food source is created

randomly using Equation 2.8. This balances the ABC algorithm's exploitation and exploration of the search space.

2.1.4 Bacterial Foraging Optimization Algorithm

The bacterial foraging optimization algorithm (BFOA) was introduced in [27]. The BFOA algorithm is inspired by the social foraging behaviour of the E. coli and M. xanthus bacteria. This theory is inspired by the principle that animals search for and obtain nutrients in a fashion that maximizes the ratio of E/T, where E is the energy obtained and T is the time spent foraging.

The BFOA's main goal is to minimize the function being used, $J(\theta)$ where θ represents the current position of a bacterium. $J(\theta)$ also represents a gradient information profile, where $J < 0$, $J = 0$, & $J > 0$ represent the presence of nutrients, a neutral position, and the presence of noxious substances, respectively [27]. Let:

$$P(j,k,l) = \{\theta^i(j,k,l)|i = 1,2,...,S\} \tag{2.11}$$

represent the positions of each member in the population of the S bacteria at the jth chemotactic step, kth reproduction step, and lth elimination-dispersal event. Let $J(i,j,k,l)$ denote the cost of the location of the ith bacterium. Nc is the length of a bacterium lifetime, measured by the number of chemotactic steps [27]. To generate a tumble, a random unit length vector is generated then added to the current position of the bacterium. This is repeated until the cost at this next position is worse than the current position being examined, or Ns, the number of chemotactic steps, is reached.

The next step of the algorithm is to calculate the interactions between bacterium in the population. Let $D_{attract}$ be the strength of the attractant released by the current bacterium, and $W_{attract}$ be the width of that attractant signal. Using local consumption, a cell will attract or repel another cell. Using $H_{repellant} = D_{attract}$ as the

height of the repellent effect and W_{repel} as the measure of the width of the repellent, then using:

$$J_{cc}(\theta) = \sum_{i=1}^{S} J_{cc}^{i} = \sum_{i=1}^{S} \left[-d_{attract} exp(-w_{attract} \sum_{j=1}^{p} (\theta_j - \theta_j^i)^2) \right] \tag{2.12}$$

$$+ \sum_{i=1}^{S} \left[h_{repellant} exp(-w_{repellant} \sum_{j=1}^{p} (\theta_j - \theta_j^i)^2 \right] \tag{2.13}$$

where $\theta = [\theta_1, ..., \theta_p]^t$ is a point in the search space domain. The bacterium will secrete attraction and repulsion chemicals affected by the environment, where a bacterium with higher nutrient concentration will secrete stronger attractant than a bacterium with a low concentration.

After N_c chemotactic steps are taken, N_{re} reproduction steps are taken. In this reproduction step, the healthiest half of the bacteria are kept while the rest are replaced with new bacteria. Lastly, the bacteria go through N_{ed} Elimination Dispersal steps, where bacteria are chosen randomly to elimination dispersal with probability P_{ed}.

2.1.5 Bat Algorithm

The bat algorithm (BA) is based on the echolocation capabilities of microbats, introduced by Yang in [46]. All microbats use echolocation to locate prey, obstacles, or other bats in their environment. For the BA some characteristics of different species of bats are generalized. These rules are:

1. All bats use echolocation to sense their distance, as well as the differences between food, prey, and their background barriers.

2. Bats will fly randomly with velocity v_i at position x_i with a fixed frequency F_{min}, varying wavelength λ and loudness A_o to search for prey [46]. The bats

can automatically adjust the frequency of their echolocation pulses, as well as the emission rate of these pulses depending on the proximity of their target [46].

3. Although loudness can vary in many ways, it is assumed that loudness varies between a large positive value A_o to a minimum constant A_{min}.

It is assumed that there is no time delay or three dimensional topography concerns because of computational complexities in multiple dimensions.

In the BA bats are moved simply, generated by the following equations:

$$f_i = f_{min} + (f_{max} - f_{min})\beta, \tag{2.14}$$

$$v_i^t = v_i^{t-1} + (x_i^t - x_*)f_i, \tag{2.15}$$

$$x_i^t = x_i^{t-1} + v_i^t \tag{2.16}$$

where $\beta \in [0, 1]$ is a random vector, x_* is the global best position of all the bats. Initial frequencies of bats are randomly chosen between $[F_{min}, F_{max}]$. A local search is performed for each bat using a random walk:

$$x_{new} = x_{old} + \epsilon A^t \tag{2.17}$$

where $\epsilon \in [-1, 1]$ is a random number and A^t is the average loudness of all the bats at this time step. The loudness A_i and rate of pulse emission R_i are updated each iteration. As a bat finds prey the loudness decreases so any value can be chosen for the A_{max} [46]. The loudness and pulse emission rate is updated as:

$$A_i^{t+1} = \alpha A_i^t \tag{2.18}$$

$$r_i^t = r_i^0 \left[1 - exp(-\gamma t)\right] \tag{2.19}$$

where α and γ are constants and t is the current iteration. [46] defines that α and γ can be the same. Bat positions, loudness, and pulse emission rate are randomly initialized [46]. The BA can be thought of as a balanced combination of the standard PSO algorithm with local search controlled by loudness and pulse rate [46].

2.1.6 Firefly Algorithm

The firefly Algorithm (FA), also introduced by Yang, is based on the flashing characteristics of fireflies [45]. Similar to BA, the following are assumptions that must be made about fireflies in the algorithm [45]:

1. All fireflies are unisex, so that one firefly will be attracted to all other fireflies regardless of their sex

2. Attractiveness is proportional to their brightness. Thus, for any two flashing fireflies, the less bright one will move towards the brighter one. If there is no brighter one than a particular firefly, it will move randomly

3. The brightness of a firefly is determined by the landscape of the objective function

In the FA, two decisions must be made: the variation of light intensity, and the formulation of the attractiveness of fireflies. The attractiveness of fireflies is determined by the evaluation of the objective function for the position of that particular firefly. The light intensity varies according to the inverse square law:

$$I(r) = I_s/r^2 \tag{2.20}$$

where I_s is the intensity at the source and r is the distance. For a given medium, the light intensity varies with

$$I = I_0 e^{-\gamma r} \tag{2.21}$$

where I_0 is the original light intensity. To avoid the singularity at $r = 0$ in equation 2.20, the combined effect of the inverse square law and absorption can be approximated as:

$$I(r) = I_0 e^{-\gamma r^2} \tag{2.22}$$

As this is proportional to a fireflies' attractiveness, the attractiveness can be defined as:

$$\beta = \beta_{min} e^{-\gamma r^2} \tag{2.23}$$

where $Beta_{min}$ is the attractiveness at $r = 0$. The distance between any two fireflies is the Cartesian Difference:

$$r_{ij} = ||x_i - x_j|| = \sqrt{\sum_{k=1}^{d} (x_{i,k} - x_{j,k})^2} \tag{2.24}$$

Lastly, the movement of a firefly i is attracted to another more attractive firefly is determined by:

$$x_i = x_i + \beta_0 e^{-\gamma r_{ij}^2} (x_j - x_i) + \alpha \epsilon_i \tag{2.25}$$

where the second term is due to the attraction of two fireflies. The parameter gamma characterizes the variation of the attractiveness thus controlling the convergence and behaviour of the FA.

2.1.7 Fish School Search Algorithm

The fish school search (FSS) algorithm is inspired by the collective behaviour of fish schools [6]. The fish in the FSS population each have an innate memory of their successes through their weights. FSS also uses the concept of evolution through a combination of collective operators which are techniques that select different modes of movement. There are two operator groups; feeding and swimming [6]. The feeding operator uses food as a metaphor for indicating the regions of the search space that

are likely to be good areas for solutions. The swimming operator is a collection of operations that attempt to guide the global search process towards areas of the search space that are sensed by the population to be more promising.

Much like real fish, the FSS population of fish are attracted to food in the search space [6]. In order to find a greater amount of food, fish in the population are able to move independently. This allows each fish to grow or shrink in weight depending on the success or failure of finding food. The weight of the fish is then updated by the following formula:

$$W_i(t+1) = W_i(t) + \frac{f\left[x_i(t+1)\right] - f\left[x_i(t)\right]}{max\{|f\left[x_i(t+1)\right] - f\left[x_i(t)\right]|\}} \tag{2.26}$$

where t is the current iteration, $Wi(t)$ is the current weight of the fish, $X_i(t)$ is the current position of the fish, and $f\left[x_i(t)\right]$ is the fitness function applied to the current position. Other important information about the weight of the fish is that the weight is updated at each iteration, there is a maximum weight W_{scale}, and all fish are initialized with weight equal to $W_{scale}/2$.

For FSS, fish swimming is directly related to all the important individual and collective behaviours such as feeding, breeding, escaping from predators, moving to more livable regions of the current habitat, or exploring socially [6]. This creates three types of movements for the FSS: individual, collective-instinct, and collective-volition. The first type of movement, individual, occurs for each fish at each iteration. This direction is randomly chosen and the fish then evaluates that position in the search space. If the position is within the bounds of the search and the food density at the new position is greater than the current position, the fish moves to this new position. This movement is also associated with a step size parameter, $Step_{ind}$, which determines the size of the movement. This parameter is decreased at each iteration.

The next movement is the collective-instinctive movement. After the individual

movement has completed, a weighted average of the individual movements of the population is calculated. The fish with success in moving individually influence the movement of the overall population more than unsuccessful fish. This movement is based on the following formula:

$$x_i(t+1) = x_i(t) + \frac{\sum_{i=1}^{N} \Delta \mathbf{x_{ind\,i}} \{f\left[x_i(t+1)\right] - f\left[x_i(t)\right]\}}{\sum_{i=1}^{N} \{f\left[x_i(t+1)\right] - f\left[x_i(t)\right]\}} \tag{2.27}$$

where $\Delta \mathbf{x_{ind\,i}}$ is the displacement of fish i due to the individual movement operator. After computation, the positions of the population are then updated.

Last is the collective-volatile movement operator. This operator is based on the overall performance of the population. It follows the following logic: if the population is putting on weight, indicating the search is successful, then the radius of the population will contract. Otherwise the radius will dilate. The collective-volatile movement operator is deemed to help greatly in enhancing the exploration abilities of the FSS. This operator is applied to each position of each fish in the population in regards to the population barycenter. The barycenter is calculated by the following formula:

$$Bari(t) = \frac{\sum_{i=1}^{N} x_i(t) W_i(t)}{\sum_{i=1}^{N} W_i(t)} \tag{2.28}$$

This movement operator will be inwards or outwards compared to the population barycenter. Fish position is then updated with the following formula.

$$x_i(t+1) = x_i(t) - step_{vol} rand(0,1)\left[x_i(t) - Bari(t)\right] \tag{2.29}$$

$$x_i(t+1) = x_i(t) + step_{vol} rand(0,1)\left[x_i(t) - Bari(t)\right] \tag{2.30}$$

If the weight of the population has increased then use 2.29, otherwise use 2.30. The $Step_{vol}$ parameter is also decreased like the $Step_{ind}$ parameter at each iteration using Equation 2.31.

$$step(t) = step_{initial} - sqrt[(1 - t^2/a^2) * b^2] \tag{2.31}$$

where a is the maximum number of iterations, b is the distance between $Step_{indinitial}$ and $Step_{indfinal}$, and t is the current iteration.

2.2 Artificial Neural Network

The artificial neural network (ANN) is a model that simulates the structure and behaviour of biological neurons. These artificial neurons are interconnected and have weights that are associated with these connections. The artificial neurons and weights are the building blocks of an ANN. ANNs are trained by weights that connect neurons. Weight updating is performed by using feedback from the error rate of the network output compared to the target output. When training is performed correctly, an ANN can be used to produce correct outputs for new unseen data.

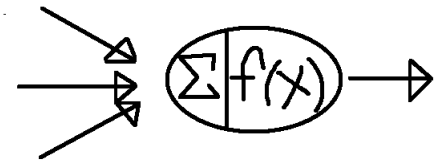

Figure 2.1: A Simple Neuron

The most basic ANN component is the artificial neuron, seen in Figure 2.1. The artificial neuron has 3 basic components: weights, a summation, and an activation function. The weights are multiplied by the input to the node. The weights represent the inherent relationship between the input data and its importance in determining

the output of the network. If the weight is positive, then data is passed further into the network. Alternatively, if the weight is negative, data is inhibited from being passed further into the network. Next, the summation sums the results of the multiplication of the inputs and the weights. This singular value is then passed through the activation function. This function must be nonlinear to allow for nonlinear relationships to be modelled. The activation function is further explained in Section 2.3.

A multi-layer ANN consists of three types of layers. The first is an input layer; this layer accepts the data as input and takes on the values of the data. There is no weight associated with the input to the input layer. Next is one or more hidden layer(s). The number of layers used is determined by the problem that needs to be solved. In this thesis, ANNs with only one hidden layer are used. The hidden layer takes the values from the input layer and follows the procedure of multiplying the connection weights by the input, summing the result, and passing the result through the activation function.

The last layer in an ANN is the output layer, which produces values that are related to the target class of the problem being passed to the ANN. This layer performs the same multiplication, summation, and application of the activation function as the hidden layer and outputs this value which is related to the task at hand. If the problem associated with this task is a classification task with two or more classes, the ANN output layer may contain a single output node that outputs values and these values must be interpreted as classes. Otherwise, the output layer could contain the same number of nodes as there are target classes (ie 10 target classes, 10 nodes in the output layer). In this case, the network would output a probability for each node in the layer that indicates how certain the model is about which class the input data is associated with. Alternatively, if this is a regression-type problem, the output layer consists of one output node that will produce a value that is related to the problem

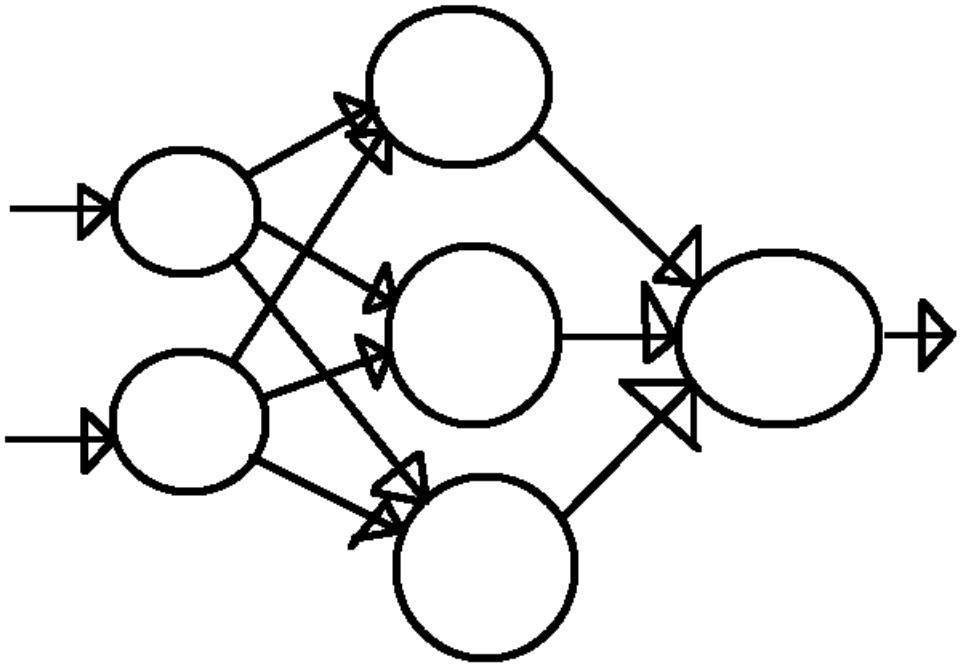

Figure 2.2: An Artificial Neural Network

being examined. For example, the network could output a predicted grade of 74.5721 for a prediction of how a student will do on a future test.

2.3 Activation Functions

The activation function of a ANN is the component that allows for a non-linear relationship to be modelled by the ANN. Some examples of activation functions can be seen in the following subsections.

2.3.1 Sigmoid Function

The sigmoid activation is defined as

$$f(net) = 1/(1 + e^{net}) \tag{2.32}$$

and has an output range of (0, 1).

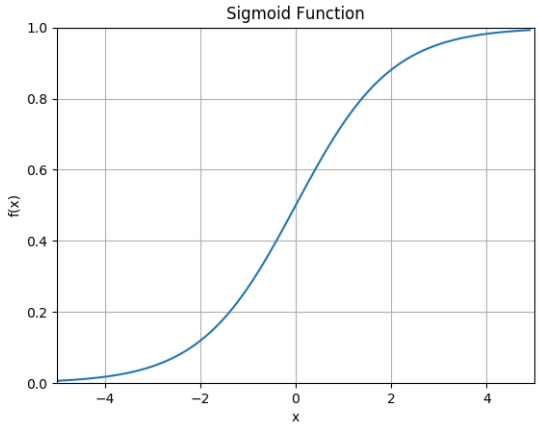

Figure 2.3: The Sigmoid function.

2.3.2 Hyperbolic Tangent Function

The Hyperbolic Tangent function, referred to as Tanh, is defined as

$$f(net) = (e^{net} - e^{-net})/(e^{net} + e^{-net}) \tag{2.33}$$

and has an output range of (-1, 1)

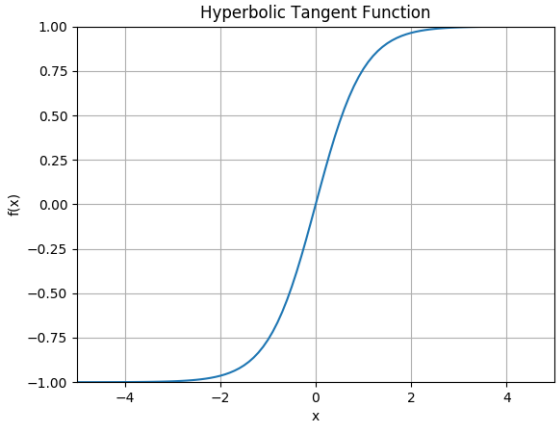

Figure 2.4: The Hyperbolic Tangent function.

2.3.3 Lecun's Hyperbolic Tangent

Lecun's Hyperbolic Tangent function, referred to as Lecun's Tanh, was suggested in [25] and is defined as

$$f(net) = 1.7159e \tanh(\frac{2}{3}net) \tag{2.34}$$

When compared to the Sigmoid function, Lecun's Tanh has a softer slope and wider output range of (-1.7159, 1.7159).

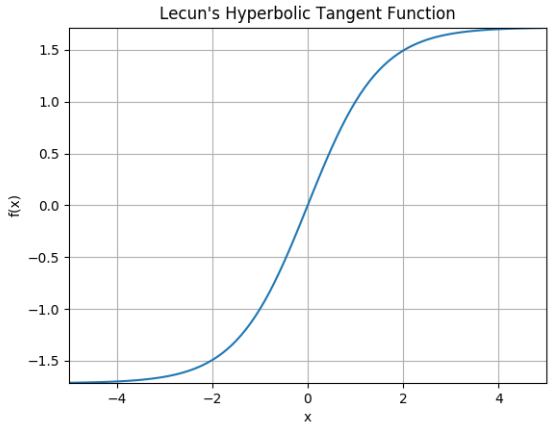

Figure 2.5: The Lecun Hyperbolic Tangent function.

2.3.4 Elliot

The Elliot activation function, hereby referred to as Elliot, is suggested in [34] and is defined as

$$f(net) = net/(1 + net) \tag{2.35}$$

This function has an output range of (-1, 1) but has a shallower gradient than Tanh, thus approaching the asymptotes slower.

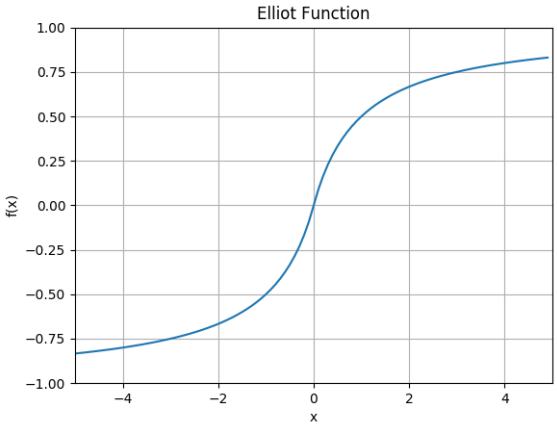

Figure 2.6: The Elliot function.

2.4 Saturation

When using bounded activation functions, it can be measured how much the hidden units in an ANN have saturated. Provided that there are enough neurons in the hidden layer, a nonlinear activation function allows the ANN to approximate any nonlinear relationship [23]. The bounds on the activation functions ensure that the signal does not grow wildly as signals propagate from layer to layer. When the input to a sigmoidal function is between the bounds of the function, the output exhibits linear behaviour. When the input to that same Sigmoidal function is outside of the bounds of the function, the output of the Sigmoidal function approaches the asymptotes, referred to as saturation. This effectively renders an ANN into a binary output state where the output of any input, with only one value outside of the activation bounds of the function, will be pushed to the asymptotic bounds of the function depending on the sign of the input.

This phenomenon is unsatisfactory in the training of an ANN since a small change in the input to a neuron will have no effect on the output of that neuron. Thus, any algorithm that attempts to change the weights of an ANN will find it difficult to evaluate whether changes to weights affects the output of the ANN, causing learning

to stall.

To further illustrate this problem, a Sigmoidal function is used in the output layer and a binary classification problem is considered. Given this, it may seem correct that saturated outputs should be found. However, in [23] it was found that a saturated output does not indicate how confident the ANN was in outputting that class. Instead, the ANN outputs the same confidence level for each example in the training set, preventing the ANN from improving on the current solution.

2.4.1 Measure of Hidden Unit Saturation

This thesis will use the measure found in [38] to measure the rate at which each algorithm saturates in the hidden layer. First, a frequency distribution is generated from the outputs of the hidden layer. This frequency distribution can then approximate the level of saturation generated by this ANN training algorithm. Next, a single valued saturation measure can be derived from the frequency distribution. The average output signal for each bin b can be calculated from the hidden layer output $g(net)$ as follows:

$$\bar{g_b} = \left\{ \begin{array}{cc} (\sum_{k=1}^{f_b g(net)_k})/f_b & if\, f_b > 0 \\ 0 & otherwise \end{array} \right\} \tag{2.36}$$

where f_b is the number of output signals in bin b. If the range of $\bar{g_b}$ is centered around 0, the absolute average will be higher for bins closer to the asymptotic values and lower for bins closer to the centre. If the range of g is $[g_L, g_U]$, gb can be scaled to -1, 1 as follows:

$$\bar{g_b}' = \frac{2(\bar{g_b} - g_L)}{g_U - g_L} - 1 \tag{2.37}$$

A weighted mean magnitude is then calculated as :

$$\varphi B = \sum_{b=1}^{B} |\bar{g_b}'| f_b \sum_{b=1}^{B} f_b \tag{2.38}$$

where B is the total number of bins, and f_b constitutes the weight of each bin. The weighted mean is the same as the arithmetic mean if the weights are equal. If the frequency was distributed uniformly in [-1, 1] then the value of φB will be 0.5. For a normal distribution in the frequency distribution the value of φB will be less than 0.5, the higher the asymptotic frequencies the closer the value of φB is to 1. In other words, as φB trends towards 1 the more saturation in the ANN. It was found in [38] that a value of 10 for the number of bins, b, is acceptable for measuring saturation.

2.5 Overfitting

The goal of ANN training is to learn the important features of the data so that the ANN can accurately classify unseen data. Typically, this is done by partitioning the dataset into three exclusive sets: training, validation, and testing. The ANN uses the training set as the data that is passed through the network to generate an output to use for Backpropagation. The validation set is used as a performance measure after every epoch of training. This set is used as an indicator of future success on previously unseen data. Lastly, the testing set is used as actual indication of success on unseen data. This set of data does not include any training example that the ANN has seen before.

Overfitting of an ANN occurs when the ANN learns to approximate the nonlinear relationship of the training data too closely. This can be described as the ANN has memorized rather than generalized, where it seems like the ANN memorized the target classes of the training set and did not generalize for the testing set. Overfitting can occur due to factors such as limited training data, too many free parameters to optimize, or too many training epochs.

2.5.1 Indication of Overfitting

Overfitting in ANN training is the phenomenon in which the ANN has adequate performance on the training dataset but poor performance on the testing dataset or any other unseen data. This can be the result of having too many free parameters which then learns the noise of the dataset. A generalization factor was developed in [39] which is an indication of overfitting behaviour of an ANN. The indicator is

$$pf = \frac{e_{test}}{e_{train}} \tag{2.39}$$

where e_{test} is the loss on the testing dataset, and e_{train} is the loss on the training dataset. It is desirable to have a $pf < 1$ which indicates that generalization error is less than the training error. A $pf > 1$ indicates that the generalization error is greater than the training error which may indicate overfitting. In this thesis pf is used to describe the overfitting behaviour of an algorithm and not as a measure of overfitting where this indicator may be able to aid in explaining behaviours of the SI trained ANNs instead of being a way to compare how overfit a network is.

Chapter 3

Swarm Intelligence Algorithms Previous Work

This is a section that describes swarm intelligence algorithm application to ANN training.

3.1 Particle Swarm Optimization

Particle swarm optimization (PSO) has been applied to the ANN training in multiple domains, including the evaluation of nonlinear functions, medical diagnoses, engineering, computer vision, and geography. In [14] it was found that PSO requires less iterations of training to achieve a similar level of error as the backpropagation algorithm. [32] applied PSO to train ANN for medical diagnoses where there was a small sample size a large number of features, and correlations between the available features. [32] found that backpropagation is generally preferred over PSO for imbalanced training data with a small number of samples and a large number of features. [47] applied PSO to adapting the ANN architecture as well as the connection weights, applying this technique to two real problems in the medical domain, finding that this technique provided good accuracy as well as good generalization ability. [33] applied PSO to a selection of classification and regression problems, such as N bit Parity, Three Color

Tube, Diabetes in Pima Indians, Sin Times Sin, and Rise Time Servomechanism. PSO was shown to be more robust when there is a high number of local minima [33]. [8] applied PSO to training ANNs while also applying PSO variants, backpropagation variants, and Hybrid approaches between PSO and backpropagation using backpropagation variants as a local search mechanism. It was shown that PSO was successful when applied to the Diabetes dataset [8]. A comparison was performed in [22] where multiple ANN training approaches were applied to four classification datasets as well as an e-Learning dataset. These approaches included PSO, Genetic Algorithm, bat algorithm, and Levenberg-Marquardt algorithm. It was found in [22] that the bat algorithm was more useful when applied to these datasets. [43] investigates the overfitting behaviour of PSO trained ANNs. [43] found that the PSO topology influenced the overfitting behaviour, as well as the use of bounded activation functions. [43] also witnessed non-convergent behaviour in the PSO swarm which was attributed to the use of bounded activation functions. When unbounded activation functions were used, it was found that the PSO swarm converged while overfitting behaviour was drastically reduced [43].

While the previously mentioned literature discusses PSO's ability to train an ANN, none of the literature attempts to discuss why this may be. [37] hypothesized that the deficiency of PSO may be due to hidden layer saturation. [37] found that while a certain degree of saturation was required for ANN success, higher levels of saturation was found to be unsatisfactory and would lead to overfitting. [43] found that non-gradient based learning can be sensitive to the degree of saturation present in an ANN. [38] devised a method to measure the degree to which an ANN has saturated that trends towards one for a more saturated ANN and zero otherwise. [38] applied four activation functions to their datasets, Sigmoid, Hyperbolic Tangent, Elliot, and Lecun's Hyperbolic Tangent, finding that Lecun's Hyperbolic Tangent function saturates the least. Through the review of the literature that has been presented here,

it can be noted that PSO suffers in training ANNs when compared with more traditional ANN learning techniques. This may be because of PSO not performing well in high dimensions, hidden unit saturation, or the overfitting behaviour.

3.2 Artificial Bee Colony Optimization

The artificial bee colony (ABC) algorithm has been applied to the training of ANNs in previous works with mild levels of success. [19] used the ABC algorithm to train ANNs for 3 benchmark functions: XOR, 3-bit parity, and 4-bit Encoder-Decoder. [19] found that the ABC algorithm produced accurate ANNs when applied to these three benchmark functions. In [20] it was found that ABC trained ANNs outperformed backpropagation trained ANNs. [40] applied ABC trained ANNs to predict overall heart function from electrocardiogram signals. [40] found that this ANN performed very satisfactorily while also reducing the time taken to train this ANN. [3] applied the ABC algorithm to train ANNs for Crime Classification. [3] found that the ABC-trained ANNs outperformed other Machine Learning algorithms, including backpropagation trained ANNs, Decision Trees, and Naive Bayes classifier. [36] used the ABC algorithm to train ANNs for modeling the daily evapotranspiration equation. [36] found that the ANNs generated were superior to the ANNs generated using the backpropagation ANNs. [41] applied the ABC-trained ANNs to predict the temperature of a volcano based on time-series data. [41] found that the basic ABC algorithm outperformed the backpropagation-trained ANNs. Lastly, [5] found that when the ABC algorithm is applied to short-term electric load forecasting for power generation planning, transmission dispatching, and day-to-day utility operations, that the ABC algorithm produces ANNs that are more suitable for this problem than PSO or Genetic Algorithms.

3.3 Ant Colony Optimization

The ant colony optimization (ACO) algorithm is, in its original state, an algorithm that is used for discrete optimization problems such as the Travelling Salesperson problem. Thus, some design decisions must be made to convert this algorithm into a discrete optimization algorithm. This conversion was first performed in [28]. At every decision point, the ant must choose a discrete point that represents a connection weight at that specific location. Once a solution was fully constructed, [28] applied backpropagation to perform a local search. [30] then applied this framework while also modifying the pheromone trail limit to allow for more search to occur as the number of iterations increases. [30] did also use backpropagation for local search once a solution was constructed. Lastly [31] built on the previous work in [30]. [31] applied ACO to nine benchmark datasets and found that ACO was superior at training ANNs for three datasets when compared to Levenberg-Marquardt, backpropagation, and ACO with backpropagation.

3.4 Bacterial Foraging Algorithm

The bacterial foraging optimization (BFO) algorithm has been used in specific applications. The first example of this is in [13] where BFO was used to create ANNs that are used to protect large power transformers. The BFO-trained ANN were then found to be a robust solution to this problem. [44] then used BFO to train ANNs to predict Alzheimer's disease, while attempting to identify the structural characteristics at baseline and over a period of two years. [44] reported an accuracy of about 92% using this approach which indicates that this is an acceptable approach.

3.5 Bat Algorithm

Initially the bat algorithm (BA) was applied in [35] in combination with backpropagation to determine the efficiency of BA when applied to training ANNs. When compared to ABC with backpropagation or backpropagation alone, [35] found that BA with backpropagation was superior. [42] applied BA to a selection of benchmark datasets where it was found that BA performed similarly to other metaheuristics, while a modified BA outperformed every other algorithm that was tested. Next [2] applied the BA to image compression where the relationship between each pixel can be nonlinear. The BA was also applied in [26] to help improve machining accuracy in thermal error modeling. [26] found that the BA with backpropagation is more stable and has higher prediction accuracy, providing a solid candidate for thermal error modeling.

3.6 Firefly Algorithm

Lastly, the firefly algorithm (FA) was applied in [17] to train an ANN to recognize characters gathered from Microsoft Paint. [17] found that the proposed FA with backpropagation technique performed better and converged quicker than other methods. Next [7] used FA to train an ANN for classification problems, namely XOR, 3-bit parity, and 4-bit Encoder-Decoder. [7] found that the ABC algorithm performed better comparatively, but FA outperformed a Genetic Algorithm. Next FA was used in [29] with backpropagation on a randomly generated dataset. [29] found that the FA did not perform well because of a lack of data, a small population of fireflies, and a low number of iterations. Lastly [15] used FA in conjunction with backpropagation for solving hydrogeneration predictions. [15] found that this approach resulted in better ANNs in terms of speed of ANN training and accuracy of predictions.

3.7 Problem Definition

As noted in Section 3.1, PSO suffers from hidden unit saturation and has some overfitting behaviour. While other SI algorithms have been applied to ANN training, no work has been done to compare the effectiveness of these other algorithms with performance, hidden unit saturation, and overfitting behaviour in mind. This thesis aims to compare the performance of SI algorithms with these metrics in mind.

Chapter 4

Implementation

This is a section that examines the implementation details of each algorithm. Each algorithm in this thesis was implemented using the Python 3.6.3 programming language [1]. Each algorithm used a population size of 50 for 1000 iterations, except for the BFO algorithm which uses a variable number of iterations depending on the problem. The fitness function of each algorithm was a forward pass of an ANN. This forward pass involved taking the position of the current solution as a parameter, splitting the solution into weights and biases, and then performing the resulting multiplications to generate a result for that solution.

The fitness of each algorithm was set to be a forward pass of data through an ANN by converting a solution's position into an ANN. If the dataset was classification based, then the fitness was the Cross-Entropy loss. If the dataset was regression based, the fitness was the Mean-Squared Error loss.

4.1 Particle Swarm Optimization

For the PSO implementation, a fully connected topology was used that shared global best information among each of the members of the population. The velocity of each particle was restricted to the range $[-1, 1]$ as performed in [38] to reduce the divergence of the PSO swarm and the synchronous version of PSO was used. The

basic algorithm can be seen in Algorithm 1.

Algorithm 1 Particle Swarm Optimization Algorithm

```
while Generation < 1000 do
  for Particles in Population do
    Evaluate Fitness
    Update Personal Best
  end for
  Update Global best
  for Particles in Population do
    Update Velocity
    Update Particle Position
  end for
end while
```

4.2 Artificial Bee Colony

The implementation of the ABC algorithm is based on [18]. This implementation represented food sources as solutions, encapsulating the number of trials, current position, and current fitness inside of this food source. This algorithm can be found in Algorithm 2.

Algorithm 2 Artificial Bee Colony Algorithm

```
while Generation < 1000 do
  Each Employed bee goes to it's food source and searches locally using Equation
  2.9
  if Fitness(Current Food Source) > Fitness(New Food Source) then
    Keep the new found food source
  end if
  Each Onlooker Bee then probabilistically chooses a food source to search around
  for better food source(s) using Equation 2.10
  if Fitness(Current Food Source) > Fitness(New Food Source) then
    Keep the new found food source
  end if
  Scout Bees will search for new food sources once a previous food source is depleted
  using Equation 2.8
end while
```

4.3 Ant Colony Optimization

Next the ACO algorithm was implemented. This implementation is based on [31] where 30 discrete points are used for each connection weight, the pheromone trails evaporate at a certain rate. The basic flow of ACO can be seen in Algorithm 3.

Algorithm 3 Ant Colony Optimization

```
Initialize Pheromone table using Equation 2.3
while Generation < 1000 do
    Probabilistic Solution Construction where an Ant chooses a value for each node
    in the ANN using Equation 2.4
    Evaluate Fitness of the Generated ANN
    Find the Best Solution
    Update Pheromone values using Best Solution using Equations 2.5 & 2.6
    Apply Pheromone Evaporation using 2.7
end while
```

4.4 Bat Algorithm

The BA implementation is directly related to the pseudocode from [46] as well as a Matlab implementation that can be found in [45]. This algorithm can be found in Algorithm 4.

4.5 Bacterial Foraging Optimization

The BFOA implementation is based on the description from the original paper [27]. This algorithm can be found in Algorithm 5.

4.6 Fish School Search

The FSS algorithm was implemented based on [6]. [16] proposed that both the Individual and Volatile step sizes decrease non-linearly to allow for the areas around

Algorithm 4 Bat Algorithm

```
Initialize Population, Velocity, Frequency, Pulse Rate, and Loudness
while Generation < 1000 do
  Move bats using Equations 2.14, 2.15, & 2.16 generating new solutions
  Evaluate Fitness of Bats
  for Bats in Population do
    if rand() > Pulse Rate of Bat then
      Generate a solution around the selected bat
    end if
  end for
  for Bats in Population do
    Generate a new Solution by Flying randomly using Equation 2.17 around current bat
    if random() < Loudness of Current Bat & Fitness(Current) > Fitness(New Solution) then
      Accept the new solution
      Decrease Loudness using Equation 2.18 & Increase Pulse Rate using Equation 2.19
    end if
  end for
  Find current Best Bat
end while
```

the global minimum to be searched in more detail while also potentially speeding up the convergence to the global minimum. The formula for this decrease is found in Equation 2.31. This algorithm can be found in Algorithm 6.

4.7 Firefly Algorithm

The Firefly algorithm was implemented based on the Matlab code provided by Xin She Yang in [45].This algorithm can be found in Algorithm 7.

Algorithm 5 Bacterial Foraging Optimization Algorithm

```
for l = 0 to N_ed do
  for k = 0 to N_re do
    for j = 0 to N_c do
      Apply a random vector to the current position
      Calculated cell to cell interactions using Equations 2.12 & 2.13
      if Fitness(Current) < Fitness(Current + Random) then
        Break
      end if
    end for
    Update best cell found
  end for
  Sort Population by Fitness
  Eliminate worst half of population
  for Cells in population do
    if rand < P_ed then
      Create new cell at random location
    end if
  end for
end for
```

Algorithm 6 Fish School Search

```
while Generations < 1000 do
  for Each Fish in Population do
    Move Fish individually by applying a random vector
    Evaluate Fitness
    Update weight of fish with Equation 2.26
  end for
  Apply Collective-Instinctive Operator which is a weighted average of individual
  movements
  Shrink or Expand the radius of the Population using Equations 2.28, 2.29, &
  2.30 based on Individual Fish Success
  Decrease Step_ind & Step_vol using Equation 2.31
end while
```

Algorithm 7 Firefly Algorithm

```
while Generation < 1000 do
  for i = 1 : NumberOfFireflies do
    for j = 1 : i do
      if I_j > I_i then
        Vary attractiveness with distance r via exp(−γR)
        Move firefly i towards j
        Evaluate new solution and update light intensity I
      end if
    end for
  end for
end while
```

Chapter 5

Experimental Setup

This is a section that describes the experiments to be conducted.

5.1 Neural Network Architectures

In this thesis, only ANNs with 3 layers are used; one input layer, one hidden layer, and one output layer. The input layer is how data is passed through to the network, where the number of input nodes is equal to the number of attributes of the dataset. The output layer of the ANN depends on the type of dataset being used, and the number of output classes in the dataset. The number of nodes in the output layer for a classification problem is equal to the number of classes present in the dataset. The number of nodes in the output layer for a regression problem is equal to the number of target outputs in the dataset. The hidden layer contains any number of hidden nodes. At the time of writing, there is not a singular best way to obtain this number of hidden nodes. In this thesis, the number of hidden neurons for some datasets were taken from literature, as indicated by the citation in Table 5.1. If the number of hidden neurons could not be found in any literature, a Genetic Algorithm was used.

Dataset	Number of Neurons
Iris [11]	4 [38]
Soybean [11]	6 [31]
Facebook [11]	100
Seizures [11]	30
Forest Fires [11]	100
Musk V2 [11]	70
Auto [11]	50
Computer Hardware [11]	40
Glass [11]	9 [38]
Spambase [11]	30
Servo [11]	50
Residential [11]	100
Parkinsons [11]	20
Music [11]	40
Breast Cancer [11]	6 [31]
Sonar (Connectionist) [11]	30
Thyroid [11]	6 [31]
Scadi [11]	20
Wine [11]	10 [43]
Ionosphere [11]	5
Zoo [11]	10
Land Cover [11]	200
MNIST [24]	100

Table 5.1: Neural Network Architectures for each Dataset

5.1.1 Defining Number of Hidden Neurons

The method of optimizing the Number of Hidden Neurons in this thesis is to use a Genetic Algorithm. The Genetic Algorithm used in this thesis was implemented using the Keras ANN framework. Keras is a machine learning framework designed for easy prototyping of ANN models written in Python. The Genetic Algorithm would generate potential ANNs, train and test the ANN, and then new ANNs would be tested through the Genetic Algorithm operators of Crossover, Selection, and Mutation. Solutions in a Genetic Algorithm are represented by chromosomes. Each chromosome is a an object with information about the ANN: the number of hidden nodes, the learning rate, and the activation function. The number of hidden nodes were set in increments of five, ranging from five to a maximum of 500. The fitness of these chromosomes was set to be the loss of the network that the chromosome generates. In classification tasks, Cross Entropy loss was used while Mean Squared Error was used in regression tasks.

With this Chromosome representation, crossover between chromosomes was a random choice between the two selected parents for each value available. Tournament Selection with $k = 4$ was used. In this Selection mechanism, four potential parents are chosen from the current population. These parents are then compared to find the one with the lowest fitness. The chromosome with the lowest fitness is then added to the pool of potential parents. Random pairs of chromosomes are chosen from this pool of parents to mate and generate the next two new chromosomes which become the next generation of chromosomes.

With the Genetic Algorithm approach, the number of hidden neurons for each dataset was tested to find an acceptable number of hidden neurons based on the performance of the network when trained using Backpropagation. This structure was then used in for every test with that dataset. The number of hidden neurons used for each dataset can be seen in the following table.

5.2 Parameter Tuning for each experiment

Bayesian Optimization (BO) was used to tune the parameters for each experiment. BO is a method of optimizing an objective function that are computationally expensive to evaluate [12]. BO consists of two components, a Bayesian statistical model for modeling the objective function and an acquisition function for deciding which parameters to test with next [12]. The statistical model is a Gaussian process which provides a Bayesian posterior probability distribution that describes potential values for some function f(x) at point x [12]. After each evaluation of f(x), the posterior distribution is updated so good values of x can be found [12]. In this thesis, the function f(x) is a swarm intelligence algorithm and x is the parameters of the current algorithm. For each combination of SI algorithm, activation function, and dataset, BO is used to tune the parameters of the algorithm. The swarm intelligence algorithms in this thesis were implemented in Python so the BO was performed using GPyOpt. GPyOpt is an open-source library for BO developed by the Machine Learning group of the University of Sheffield [4]. This package boasts capabilities of automatic configuration of models and Machine Learning algorithms, parallel experiments, and mixed types of variables [4].

5.3 Experimental Parameters

The GPyOpt package was applied to tune the parameters for each combination of algorithm, activation function, and dataset. The parameters were tuned in the following ranges. The following tables contain the parameters that were used for each algorithm, activation function, dataset combination. An example of these parameters are found in Table 5.2 with the remaining parameters found in Section A.

Dataset	W	C1	C2
Auto	0.42434	1.82755	1.58621
BreastCancer	0.60270	1.89159	1.13346
ComputerHardware	0.77202	1.37043	1.51437
Facebook	0.79332	1.72871	1.46039
Forest Fires	0.56748	1.77542	1.09125
Glass	0.63589	1.37756	1.00825
Ionosphere	0.90000	1.00000	1.00000
Iris	0.49452	1.57499	1.98687
LandCover	0.68048	1.22115	1.17707
MNIST	0.40000	1.74774	1.91942
Music	0.70335	1.46544	1.97039
Musk	0.57469	1.73502	1.69297
Parkinsons	0.44225	1.42556	1.51641
Residential	0.90000	2.00000	2.00000
Scadi	0.40000	1.51375	1.00000
Seisures	0.40000	1.92354	1.92764
Servo	0.55772	1.30412	1.68724
Sonar	0.69614	1.54019	1.92433
Soybean	0.71224	1.51943	1.50465
Spambase	0.40000	1.80524	2.00000
Thyroid	0.52957	2.00000	1.90449
Wine	0.90000	1.00000	2.00000
Zoo	0.69983	1.54484	1.87629

Table 5.2: PSO with Tanh Activation function parameters

Chapter 6

Results

This section describes the results found after performing the experiments. The first section describes results based on only the regression datasets, the second section only the classification datasets, the third section focuses on overall results, the fourth section examines the hidden unit saturation of each algorithm and activation function combination, while the last section describes the overfitting behaviours of each of the algorithms. Both the regression and classification sections will examine the results from two perspectives; training and testing, while the classification section will also include accuracy and loss results. The regression section will only focus on the loss results. Rankings for all algorithms were performed using the Wilcoxon rank sum test as described in [9]. Using this method, all algorithms can be given a rank based on how they perform on average across all activation functions on a given dataset. This ranking can then be averaged to generate an overall average rank which is used to rank the algorithm for an overall rank.

Lastly the results will be examined through the size of the dataset that was being used. The datasets will be split into small, medium, and large. Small datasets will have less than 500 instances or less than 50 features. Medium datasets will have more than 500 instances or more than 50 features. Large datasets will have more than 10,000 instances or more than 500 features. The small datasets are: Auto, Glass,

Iris, Parkinson's, Wine, Zoo, Computer Hardware, Ionosphere, Servo, and Soybean. Medium datasets are: Thyroid, Breast Cancer, Facebook, Forest Fires, Land Cover, Music, Musk, Sonar, Residential, Scadi, Seizures, Spambase. Lastly, the large dataset is MNIST.

6.1 Regression Results

First, the testing loss of the regression datasets must be examined. These results, shown in Table 6.1, are similar to the training loss results. Much like the training loss, PSO generated the best networks for testing loss based on the rank and the number of times that PSO generated the network that was ranked first for a dataset. Again, the BA followed PSO in second, with BFA & ACO moving up to third. Next it was observed that FSS remained in fifth, ABC moved down a rank to sixth, with Firefly remaining in seventh. It would be worthwhile to note that PSO did not generate all of the first ranked ANNs when examining the First Rank frequency. When examining the testing loss rankings it can be seen that BA produced two of the top performing networks, while BFA and FSS both produced one. This could be due to either hidden unit saturation or overfitting but will be examined in more depth in Section's 6.4 & 6.5. For now, it can be said that when using an SI algorithm for a regression problem that PSO would be the most robust algorithm to use based on the performance shown in this section.

6.2 Classification Results

Next the performance of the SI algorithms in regards to loss will be examined. The firefly algorithm was found to be the top performer in this set of rankings, producing six of the top generated ANNs. Following Firefly was ACO, with one top ANN generated, FSS in third while producing five top ANNs, PSO in fourth with four top ANNs,

Algorithm	PSO	ABC	ACO	BA	BFA	Firefly	Fish
Auto	1	4	2	3	5	7	6
ComputerHardware	1	6	7	1	1	1	5
Facebook	2	5	4	1	3	7	5
ForestFires	1	5	2	3	6	7	4
Music	1	6	4	2	3	7	4
Residential	1	7	4	3	2	6	5
Servo	1	5	3	2	6	7	4
Average Rank	1.1429	5.4286	3.7143	2.1429	3.7143	6	4.714
Ranking	1	6	3	2	3	7	5
First Frequency	6	0	0	2	1	1	0

Table 6.1: Testing Loss Regression Datasets

ABC in fifth, BA in sixth, and BFA in seventh; each with one top ANN generated. This is the first time in this thesis that PSO was not the top ranked algorithm. This ranking indicates that there may be better algorithms to use for training ANNs for classification than PSO. This will be further examined when examining the accuracy generated by these same networks.

Lastly the accuracy of the ANNs must be examined. These results can be found in Table 6.3. For the second time in the classification results, Firefly was ranked first as the top algorithm with seven first place ranks. Next was ACO in second with four top ranks, third was PSO with six top ranks, followed by ABC and FSS tied for fourth with four and five top ranks a piece. Next came BA in sixth with two top ranks, while BFA finished ranked seventh with two top generated ANNs. ACO out-ranking PSO is an important result.

6.3 Overall Results

Next the testing loss of the SI trained ANNs are observed, found in Table 6.4. These rankings show that PSO again is the top ranked algorithm for ANN training from a loss perspective. This result does follow some of the results from the previous sections. First the number of top ranked ANNs that PSO generates is 10. Next, it can

Algorithm	PSO	ABC	ACO	BA	BFA	Firefly	Fish
BreastCancer	1	6	7	1	1	1	5
Glass	6	2	3	4	7	5	1
Ionosphere	6	2	4	3	7	5	1
Iris	7	3	6	3	5	2	1
LandCover	4	6	2	3	7	1	5
MNIST	2	6	2	4	7	1	5
Musk	2	6	3	4	7	1	5
Parkinsons	5	1	2	6	7	4	3
Scadi	6	3	1	5	7	2	3
Seizures	1	7	6	3	5	2	4
Sonar	5	3	2	4	7	1	6
Soybean	2	3	4	4	7	1	6
Spambase	1	4	3	5	7	2	5
Thyroid	1	3	6	5	7	2	4
Wine	5	4	2	6	7	3	1
Zoo	5	3	2	4	7	5	1
Average Rank	3.6875	3.875	3.4375	4	6.375	2.375	3.5
Ranking	4	5	2	6	7	1	3
First Frequency	4	1	1	1	1	6	5

Table 6.2: Testing Loss Classification Datasets

be observed that BA finished as the second ranked algorithm with three top ranked ANNs generated. Following BA, Firefly which generated the second highest number of first ranked ANNs with seven, but the performance, last in loss, on the regression datasets hampers the firefly algorithm's average ranking in this case. Ranked fourth is ACO. After ACO is FSS which generated five top ranking ANNs. Ranked sixth overall is ABC with one top ANN produced, followed by BFA in seventh with two top ANNs.

The results broken down by dataset size can be seen in Tables 6.5, 6.6, and 6.7 for the loss ranks, or in Tables 6.8, 6.9, and 6.10 for the accuracy ranks. Both of these sets of Tables don't contain anything telling about the performance of the algorithms based on dataset size. These results follow the overall results fairly consistently.

Algorithm	PSO	ABC	ACO	BA	BFA	Firefly	Fish
BreastCancer	1	1	1	1	1	1	1
Glass	4	1	6	5	7	3	1
Ionosphere	1	1	1	1	1	1	1
Iris	7	5	3	6	4	2	1
LandCover	3	6	2	5	7	1	4
MNIST	1	6	2	4	7	3	5
Musk	2	6	3	4	7	1	5
Parkinsons	4	1	3	6	7	2	5
Scadi	2	5	1	6	7	3	4
Seizures	1	5	4	3	7	2	5
Sonar	3	4	2	5	7	1	6
Soybean	1	4	4	3	7	1	6
Spambase	1	4	3	6	7	2	4
Thyroid	7	4	1	3	2	6	4
Wine	3	3	2	6	7	1	5
Zoo	3	2	3	6	7	5	1
Average	2.75	3.625	2.5625	4.375	5.75	2.1875	3.625
Ranks	3	4	2	6	7	1	4
First Frequency	6	4	4	2	2	7	5

Table 6.3: Testing Accuracy Classification Datasets

Algorithm	PSO	ABC	ACO	BA	BFA	Firefly	Fish
Auto	1	4	2	3	5	7	6
BreastCancer	1	6	7	1	1	1	5
ComputerHardware	1	6	7	1	1	1	5
Facebook	2	5	4	1	3	7	5
ForestFires	1	5	2	3	6	7	4
Glass	6	2	3	4	7	5	1
Ionosphere	6	2	4	3	7	5	1
Iris	7	3	6	3	5	2	1
LandCover	4	6	2	3	7	1	5
MNIST	2	6	2	4	7	1	5
Music	1	6	4	2	3	7	4
Musk	2	6	3	4	7	1	5
Parkinsons	5	1	2	6	7	4	3
Residential	1	7	4	3	2	6	5
Scadi	6	3	1	5	7	2	3
Seizures	1	7	6	3	5	2	4
Servo	1	5	3	2	6	7	4
Sonar	5	3	2	4	7	1	6
Soybean	2	3	4	4	7	1	6
Spambase	1	4	3	5	7	2	5
Thyroid	1	3	6	5	7	2	4
Wine	5	4	2	6	7	3	1
Zoo	5	3	2	4	7	5	1
Average	2.9130	4.3478	3.5217	3.4348	5.5652	3.4783	3.8696
Ranks	1	6	4	2	7	3	5
First Frequency	10	1	1	3	2	7	5

Table 6.4: Testing Loss Overall Results

	PSO	ABC	ACO	Bat	BFA	Firefly	Fish
Auto	1	4	2	3	5	7	6
ComputerHardware	1	6	7	1	1	1	5
Glass	6	2	3	4	7	5	1
Ionosphere	6	2	4	3	7	5	1
Iris	7	3	6	3	5	2	1
Parkinsons	5	1	2	6	7	4	3
Servo	1	5	3	2	6	7	4
Soybean	2	3	4	4	7	1	6
Wine	5	4	2	6	7	3	1
Zoo	5	3	2	4	7	5	1
Average rank	3.9	3.3	3.5	3.6	5.9	4	2.9
Rank	5	2	3	4	7	6	1
First Frequency	3	1	0	1	1	2	5

Table 6.5: Testing Loss Small Dataset Ranks

	PSO	ABC	ACO	Bat	BFA	Firefly	Fish
BreastCancer	1	6	7	1	1	1	5
Facebook	2	5	4	1	3	7	5
ForestFires	1	5	2	3	6	7	4
LandCover	4	6	2	3	7	1	5
Music	1	6	4	2	3	7	4
Musk	2	6	3	4	7	1	5
Residential	1	7	4	3	2	6	5
Scadi	6	3	1	5	7	2	3
Seisures	1	7	6	3	5	2	4
Sonar	5	3	2	4	7	1	6
Spambase	1	4	3	5	7	2	5
Thyroid	1	3	6	5	7	2	4
Average rank	2.17	5.08	3.67	3.25	5.17	3.25	4.58
Rank	1	6	4	2	7	2	5
First Frequency	6	0	1	0	0	3	0

Table 6.6: Testing Loss Medium Dataset Ranks

	PSO	ABC	ACO	Bat	BFA	Firefly	Fish
MNIST	2	6	2	4	7	1	5

Table 6.7: Testing Loss Large Dataset Ranks

	PSO	ABC	ACO	Bat	BFA	Firefly	Fish
Glass	4	1	6	5	7	3	1
Ionosphere	1	1	1	1	1	1	1
Iris	7	5	3	6	4	2	1
Parkinsons	4	1	3	6	7	2	5
Soybean	1	4	4	3	7	1	6
Wine	3	3	2	6	7	1	5
Zoo	3	2	3	6	7	5	1
Average rank	3.29	2.43	3.14	4.71	5.71	2.14	2.86
Rank	5	2	4	6	7	1	3
First Frequency	2	3	1	1	1	3	4

Table 6.8: Testing Accuracy Small Dataset Ranks

	PSO	ABC	ACO	Bat	BFA	Firefly	Fish
BreastCancer	1	1	1	1	1	1	1
LandCover	3	6	2	5	7	1	4
Musk	2	6	3	4	7	1	5
Scadi	2	5	1	6	7	3	4
Seisures	1	5	4	3	7	2	5
Sonar	3	4	2	5	7	1	6
Spambase	1	4	3	6	7	2	4
Thyroid	7	4	1	3	2	6	4
Average rank	2.50	4.38	2.13	4.13	5.63	2.13	4.13
Rank	3	6	1	4	7	1	4
First Frequency	3	1	3	1	1	4	1

Table 6.9: Testing Accuracy Medium Dataset Ranks

	PSO	ABC	ACO	Bat	BFA	Firefly	Fish
MNIST	2	6	2	4	7	1	5

Table 6.10: Testing Accuracy Large Dataset Ranks

6.4 Saturation Results

This section describes the results of measuring the amount to which each algorithm caused the ANN hidden units to saturate. The measure used to describe this saturation can be found in Section 2. The ranking procedure was performed directly on the resulting values generated from the saturation measure, as each measure was scaled in the same $[-1, 1]$ range with a ranking of one being the lowest saturating network generated.

Algorithm/Dataset	PSO	ABC	ACO	BA	BFA	Firefly	Fish
Auto	3	2	4	6	7	5	1
ComputerHardware	5	4	2	7	6	1	3
Facebook	5	3	1	7	6	2	4
ForestFires	6	3	2	5	7	1	4
Music	5	2	1	7	6	4	3
Residential	5	1	3	7	6	4	2
Servo	5	2	1	3	7	3	6
Average Rank	4.8571	2.4286	2	6	6.4286	2.8571	3.2857
First Frequency	0	1	3	0	0	2	1

Table 6.11: Saturation Rankings for Regression Datasets

When examining the results found in Table 6.13 it can be found that ACO & ABC were the top ranking algorithms, while producing 11 & 1, respectively, of the top ranked saturation measures. Next it was observed that FSS was ranked third with three top ranked ANNs, followed by Firefly with eight top ranked ANNs. The BA was observed to be ranked fifth but did not produce any top ranking ANNs, with PSO ranking sixth and also not generating any top ranked ANNs. Lastly, BFA was found to be ranked seventh while failing to generate any top ranked ANN.

Finding ACO as the top ranking algorithm based on saturation was expected. The ACO algorithm used in this thesis did not allow for weights for the ANN to come from outside of the bounds of the activation function. For example, the points randomly generated for each weight in the ANN with the Tanh activation function come from within the range $[-1, 1]$. Therefore any performance that the ACO had is somewhat

Algorithm/Dataset	PSO	ABC	ACO	BA	BFA	Firefly	Fish
BreastCancer	4	2	1	6	7	5	3
Glass	7	5	1	6	4	3	2
Ionosphere	6	3	1	4	7	5	2
Iris	4	2	1	6	7	4	3
LandCover	3	2	7	4	6	1	5
MNIST	7	4	5	3	6	2	1
Musk	7	2	5	6	4	1	3
Parkinsons	6	2	1	5	4	7	3
Scadi	5	2	6	3	7	1	4
Seisures	7	2	5	3	6	1	3
Sonar	7	2	5	2	6	1	2
Soybean	6	3	2	7	4	5	1
Spambase	6	2	2	5	7	1	4
Thyroid	7	3	1	5	6	4	2
Wine	6	3	1	5	7	4	2
Zoo	4	3	1	5	7	6	2
Average Rank	5.75	2.625	2.8125	4.6875	5.9375	3.1875	2.625
First Frequency	0	0	8	0	0	6	2

Table 6.12: Saturation Results for Classification Datasets

hidden by the fact that it does not saturate much. It needs to be examined whether or not ACO generates ANNs that saturate for problems that it performs well on. For this, Table 6.14 should be examined. From this table, it can be seen that a saturation rank between one and three produced 10 training loss ranks of below a rank of four, which is in the bottom half of all algorithms tested. A high rank for saturation also produced a training loss rank of above four, four times. Nine times the saturation rank was below four and ACO produced high training loss ranks in eight of those datasets. This comparison can also be extended to testing loss. Of the 14 cases of a high ranking for saturation, ACO produced six testing loss ranks that are below four. For testing loss ACO produced five testing loss ranks above four when the saturation rank was below four. This comparison is used to show that some level of saturation is needed for ANN training. The cases where ACO did not saturate produced lower performing ANNs based on training and testing loss. In contrast to those cases, the

Algorithm	PSO	ABC	ACO	BA	BFA	Firefly	Fish
Auto	3	2	4	6	7	5	1
BreastCancer	4	2	1	6	7	5	3
ComputerHardware	5	4	2	7	6	1	3
Facebook	5	3	1	7	6	2	4
ForestFires	6	3	2	5	7	1	4
Glass	7	5	1	6	4	3	2
Ionosphere	6	3	1	4	7	5	2
Iris	4	2	1	6	7	4	3
LandCover	3	2	7	4	6	1	5
MNIST	7	4	5	3	6	2	1
Music	5	2	1	7	6	4	3
Musk	7	2	5	6	4	1	3
Parkinsons	6	2	1	5	4	7	3
Residential	5	1	3	7	6	4	2
Scadi	5	2	6	3	7	1	4
Seizures	7	2	5	3	6	1	3
Servo	5	2	1	3	7	3	6
Sonar	7	2	5	2	6	1	2
Soybean	6	3	2	7	4	5	1
Spambase	6	2	2	5	7	1	4
Thyroid	7	3	1	5	6	4	2
Wine	6	3	1	5	7	4	2
Zoo	4	3	1	5	7	6	2
Average Rank	5.4783	2.5652	2.5652	5.0870	6.0870	3.0870	2.8261
Rank	6	1	1	5	7	4	3
First Frequency	0	1	11	0	0	8	3

Table 6.13: Saturation Results

Dataset	Saturation Rank	Training Loss Rank	Testing Loss
BreastCancer	1	7	7
Glass	1	6	3
Ionosphere	1	6	4
Iris	1	7	6
LandCover	7	1	2
MNIST	5	3	2
Musk	5	2	3
Parkinsons	1	4	2
Scadi	6	1	1
Seisures	5	4	6
Sonar	5	2	2
Soybean	2	5	4
Spambase	2	2	3
Thyroid	1	6	6
Wine	1	6	2
Zoo	1	5	2
Auto	2	4	2
ComputerHardware	6	2	7
Facebook	2	1	4
ForestFires	2	2	2
Music	4	1	4
Residential	4	3	4
Servo	3	1	3

Table 6.14: ACO Saturation Results Compared to Training and Testing Loss

cases where ACO did saturate the ANNs produced both ANNs that perform well and ones that don't perform well.

6.5 Overfitting Results

It should be noted again that the results in Table 6.15 do not measure any overfitting that may or may not have occurred. The results in Table 6.15 are simply an indication of whether it is possible that overfitting has occurred. The training data used for this measure is included in Section A.

As can be seen in the table, PSO had the worst rank in when applying the overfitting indicator. In fact in all but one of the datasets PSO finished fourth or worse, with

the only dataset that it ranked first in being the one that all algorithms were ranked first. The sixth ranked algorithm is ACO which only ranked first in two datasets. The next algorithm is firefly algorithm, which generated five ANNs ranked first. ABC and BA tied for the third rank, with two and one top ranked ANNs generated respectively. The second ranked algorithm is FSS which produced three top ranked ANNs. Lastly, BFA was ranked as the algorithm that indicated overfitting the least, producing 16 top ranked ANNs.

The result of PSO having the worst ranking based on overfitting indicator falls in line with the previous research in this area. When examining the SI algorithms' results in this thesis, it was noted that there were algorithms that outperformed PSO in terms of loss and accuracy, namely ACO, Firefly, and BA.

	PSO	ABC	ACO	BA	BFA	Firefly	Fish
Auto	7	3	6	5	4	2	1
Breast Cancer	1	1	1	1	1	1	1
Computer Hardware	6	4	2	7	5	1	3
Facebook	4	5	7	3	1	2	6
Forest Fires	7	5	6	4	2	1	3
Glass	5	7	2	4	1	6	3
Ionosphere	6	5	2	4	1	7	3
Iris	7	5	1	4	3	6	2
LandCover	6	4	7	2	1	5	3
MNIST	7	1	6	4	1	5	3
Music	7	3	4	6	5	2	1
Musk	7	4	6	3	1	5	2
Parkinsons	6	4	5	2	1	7	3
Residential	7	3	6	5	2	1	4
Scadi	6	2	7	3	1	4	5
Seisures	7	4	5	3	1	6	2
Servo	7	3	5	6	4	1	2
Sonar	7	3	6	4	1	5	2
Soybean	7	4	3	5	1	6	2
Spambase	7	3	6	4	1	5	2
Thyroid	7	5	2	3	1	6	3
Wine	7	6	2	3	1	5	4
Zoo	6	5	3	4	1	7	2
Average rank	6.2609	3.8696	4.3478	3.8696	1.7826	4.1739	2.6957
Rank	7	3	6	3	1	5	2
First Frequency	1	2	2	1	16	5	3

Table 6.15: Overfitting Results

Chapter 7

Conclusion

7.1 Performance of SI Algorithms

In this work, we compared the results of using swarm intelligence algorithms to train artificial neural networks of various sizes on both classification and regression based datasets. Based on these comparisons, the following conclusions can be made:

- When applied to regression datasets, it was found that particle swarm optimization was the best performing algorithm in terms of testing loss.
- The firefly algorithm, ant colony optimization algorithm, and fish school search each outperformed particle swarm optimization in terms of testing loss on the classification datasets.
- The firefly algorithm and ant colony optimization algorithm also surpassed particle swarm optimization in testing accuracy when applied to classification datasets.
- Overall, it was found that particle swarm optimization was the algorithm that was the performed the best when applied to all datasets, ranking first overall in terms of testing loss.

- Next, it was found that the ant colony optimization algorithm was the algorithm that saturated the least because of the way in which the algorithm chooses weights of the ANN.

- The relationship between saturation and Artificial Neural Network performance was then examined using ant colony optimization as an example. It was demonstrated that saturation and network performance is not an exact inverse or one to one relationship. There are some cases where higher levels of saturation may have aided the performance and there were also cases where lower levels of saturation may have aided performance. The same is true for the decrease in performance.

- Lastly, an indication of overfitting was applied to the training and testing losses to indicate how the algorithms may have performed. This application found that the particle swarm optimization algorithm generated results that may indicate overfitting at a higher rate than the other algorithms.

- The bacterial foraging optimization algorithm was found to produce results that may indicate overfitting at the lowest rate when compared to the other algorithms but it was found that it was one of the worst performing algorithms.

7.2 Future Work

There are some areas for improvement that may improve performance in one of the metrics that was examined in this thesis. This thesis focused on the vanilla implementations of each algorithm so any future work would update the algorithm to a current variant. This update would allow for any recent developments that may improve performance to be included in the application to get a more up to date view of how each algorithm performs. Another future work option would be to include

a validation technique such as early stopping. Given the results found in this thesis it could be hypothesized that the performance of certain algorithms was due to over training which caused overfitting. With an early stopping technique the training would stop once certain criteria are met which would ideally improve performance. Another technique to improve performance would be to implement a cross-validation technique for each iteration of training. This would go hand in hand with the early stopping as using the validation set from cross validation would give a more realistic idea of testing performance throughout training. Lastly, future work could compare the computational cost of each of the algorithms or other metrics to give an even more comprehensive look into the performance of these algorithms.

Bibliography

[1] Python software foundation. python language reference, version 3.7, 2019 (Accessed October 14th, 2019).

[2] Adis Alihodzic. Training feed-forward neural networks employing improved bat algorithm for digital image compression. In *Large-Scale Scientific Computing*, pages 315–323, Cham, 2018 (Accessed June 20th, 2019). Springer International Publishing.

[3] Syahid Anuar, Ali Selamat, and Roselina Sallehuddin. Hybrid artificial neural network with artificial bee colony algorithm for crime classification. In *Computational Intelligence in Information Systems*, pages 31–40, Cham, 2015 (Accessed June 17th, 2019). Springer International Publishing.

[4] The GPyOpt authors. Gpyopt: A bayesian optimization framework in python. `http://github.com/SheffieldML/GPyOpt`, 2016 (Accessed June 5th, 2019).

[5] Shahid M. Awan, Muhammad Aslam, Zubair A. Khan, and Hassan Saeed. An efficient model based on artificial bee colony optimization algorithm with neural networks for electric load forecasting. *Neural Computing and Applications*, 25(7):1967–1978, December 2014 (Accessed June 17th, 2019).

[6] C. J. A. Bastos Filho, F. B. de Lima Neto, A. J. C. C. Lins, A. I. S. Nascimento, and M. P. Lima. A novel search algorithm based on fish school behavior. In *2008 IEEE International Conference on Systems, Man and Cybernetics*, pages 2646–2651, October 2008 (Accessed August 2nd, 2019).

[7] Ivona Brajevic and Milan Tuba. Training feed-forward neural networks using firefly algorithm. February 2013 (Accessed June 20th, 2019).

[8] M. Carvalho and T. B. Ludermir. Hybrid training of feed-forward neural networks with particle swarm optimization. In *Neural Information Processing*, pages

1061–1070, Berlin, Heidelberg, 2006 (Accessed June 9th, 2019). Springer Berlin Heidelberg.

[9] Joaqun Derrac, Salvador Garca, Daniel Molina, and Francisco Herrera. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation*, 1(1):3 – 18, 2011.

[10] Marco Dorigo. *Optimization, Learning and Natural Algorithms*. PhD thesis, 1992 (Accessed July 26th, 2019).

[11] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017 (Accessed July 20th, 2019).

[12] Peter I. Frazier. A Tutorial on Bayesian Optimization. *arXiv e-prints*, July 2018 (Accessed June 5th, 2019).

[13] M. Geethanjali, V. Kannan, and A. V. R. Anjana. Bacterial foraging optimization algorithm trained ann based differential protection scheme for power transformers. In *Swarm, Evolutionary, and Memetic Computing*, pages 267–277, Berlin, Heidelberg, 2011 (Accessed July 15th, 2019). Springer Berlin Heidelberg.

[14] V. G. Gudise and G. K. Venayagamoorthy. Comparison of particle swarm optimization and backpropagation as training algorithms for neural networks. In *Proceedings of the 2003 IEEE Swarm Intelligence Symposium. SIS'03 (Cat. No.03EX706)*, pages 110–117, April.

[15] Ali Thaeer Hammid, Mohd Herwan Bin Sulaiman, and Omar I. Awad. A robust firefly algorithm with backpropagation neural networks for solving hydrogeneration prediction. *Electrical Engineering*, 100(4):2617–2633, December 2018 (Accessed June 20th, 2019).

[16] Andreas Janecek and Ying Tan. Feeding the fish – weight update strategies for the fish school search algorithm. In *Advances in Swarm Intelligence*, pages 553–562, Berlin, Heidelberg, 2011 (Accessed June 25th, 2019). Springer Berlin Heidelberg.

[17] M K. Sahoo, Janmenjoy Nayak, S Mohapatra, B K. Nayak, and Dr. H. Behera. *Character Recognition Using Firefly Based Back Propagation Neural Network*, volume 32, pages 151–164. 01 2015 (Accessed June 20th, 2019).

[18] Dervis Karaboga. An idea based on honey bee swarm for numerical optimization. 2005 (Accessed July 29th, 2019).

[19] Dervis Karaboga, Bahriye Akay, and Celal Ozturk. Artificial bee colony (abc) optimization algorithm for training feed-forward neural networks. In *Modeling Decisions for Artificial Intelligence*, pages 318–329, Berlin, Heidelberg, 2007 (Accessed June 17th, 2019). Springer Berlin Heidelberg.

[20] Asha Gowda Karegowda and Manjunath Darshan. Optimizing feed forward neural network connection weights using artificial bee colony algorithm. 2013 (Accessed June 17th, 2019).

[21] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proceedings of ICNN'95 - International Conference on Neural Networks*, volume 4, pages 1942–1948, November 1995 (Accessed July 24th, 2019).

[22] Koffka Khan and Sahai Ashok. A comparison of ba, ga, pso, bp and lm for training feed forward neural networks in e-learning context. *International Journal of Intelligent Systems and Applications*, 4, June 2012 (Accessed June 10th, 2019).

[23] Steve Lawrence, Ah Chung Tsoi, and Andrew D. Back. Function approximation with neural networks and local methods: Bias, variance and smoothness. pages 16–21, 1996 (Accessed June 10th, 2019).

[24] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, November 1998 (Accessed July 20th, 2019).

[25] Yann LeCun, Léon Bottou, Genevieve B. Orr, and Klaus-Robert Müller. Efficient backprop. pages 9–50, 1998 (Accessed June 20th, 2019).

[26] Yang Li, Ji Zhao, and Shijun Ji. Thermal positioning error modeling of machine tools using a bat algorithm-based back propagation neural network. *The International Journal of Advanced Manufacturing Technology*, 97(5):2575–2586, July 2018 (Accessed June 20th, 2019).

[27] Y. Liu and K. M. Passino. Biomimicry of social foraging bacteria for distributed optimization: Models, principles, and emergent behaviors. *J. Optim. Theory Appl.*, 115(3):603–628, December 2002 (Accessed July 30th, 2019).

[28] Yan-Peng Liu, Ming-Guang Wu, and Ji-Xin Qian. Evolving neural networks using the hybrid of ant colony optimization and bp algorithms. In *Advances in Neural Networks*, pages 714–722, Berlin, Heidelberg, 2006 (Accessed June 20th, 2019). Springer Berlin Heidelberg.

[29] Sudip Mandal, Goutam Saha, and Rajat Pal. Neural network training using firefly algorithm. *Global Journal on Advancement in Engineering and Science*, 1:7–11, March 2015 (Accessed June 20th, 2019).

[30] M. Mavrovouniotis and S. Yang. Evolving neural networks using ant colony optimization with pheromone trail limits. In *2013 13th UK Workshop on Computational Intelligence (UKCI)*, pages 16–23, September 2013 (Accessed June 20th, 2019).

[31] Michalis Mavrovouniotis and Shengxiang Yang. Training neural networks with ant colony optimization algorithms for pattern classification. *Soft Computing*, 19(6):1511–1522, June 2015 (Accessed July 15th, 2019).

[32] Maciej Mazurowski, Piotr Habas, Jacek Zurada, Joseph Lo, Jay A Baker, and Georgia Tourassi. Training neural network classifiers for medical decision making: The effects of imbalanced datasets on classification performance. *Neural networks : the official journal of the International Neural Network Society*, 21:427–36, March 2008 (Accessed July 10th, 2019).

[33] R. Mendes, P. Cortez, M. Rocha, and J. Neves. Particle swarms for feedforward neural network training. In *Proceedings of the 2002 International Joint Conference on Neural Networks*, volume 2, pages 1895–1899, May 2002 (Accessed June 10th, 2019).

[34] The National Science Foundation and David Elliott. A better activation function for artificial neural networks. December 1998 (accessed February 3, 2014).

[35] Nazri Mohd. Nawi, Muhammad Zubair Rehman, and Abdullah Khan. A new bat based back-propagation algorithm. In *Advances in Systems Science*, pages 395–404, Cham, 2014 (Accessed June 20th, 2019). Springer International Publishing.

[36] Coskun Ozkan, Ozgur Kisi, and Bahriye Akay. Neural networks with artificial bee colony algorithm for modeling daily reference evapotranspiration. *Irrigation Science*, 29(6):431–441, November 2011 (Accessed June 17th, 2019).

[37] A. Rakitianskaia and A. Engelbrecht. Saturation in pso neural network training: Good or evil? In *2015 IEEE Congress on Evolutionary Computation (CEC)*, pages 125–132, May 2015 (Accessed June 10th, 2019).

[38] A. Rakitianskaia and A. Engelbrecht. Measuring saturation in neural networks. In *2015 IEEE Symposium Series on Computational Intelligence*, pages 1423–1430, December 2015 (Accessed June 28th, 2019).

[39] A. Rbel and Technische Universitat Berlin. The dynamic pattern selection algorithm: Effective training and controlled generalization of backpropagation neural networks, 1994 (Accessed June 13th, 2019).

[40] Slami Saadi, Maamar Bettayeb, Abderrezak Guessoum, and M. K. Abdelhafidi. Artificial bees colony optimized neural network model for ecg signals classification. In *Neural Information Processing*, pages 339–346, Berlin, Heidelberg, 2012 (Accessed June 17th, 2019). Springer Berlin Heidelberg.

[41] Habib Shah, Rozaida Ghazali, and Nazri Mohd Nawi. Hybrid ant bee colony algorithm for volcano temperature prediction. In *Emerging Trends and Applications in Information Communication Technologies*, pages 453–465, Berlin, Heidelberg, 2012 (Accessed June 17th, 2019). Springer Berlin Heidelberg.

[42] Milan Tuba, Adis Alihodzic, and Nebojsa Bacanin. *Cuckoo Search and Bat Algorithm Applied to Training Feed-Forward Neural Networks*, pages 139–162. Springer International Publishing, 2015 (Accessed June 20th, 2019).

[43] A. B. van Wyk and A. P. Engelbrecht. Overfitting by pso trained feedforward neural networks. In *IEEE Congress on Evolutionary Computation*, pages 1–8, July 2010 (Accessed June 11th, 2019).

[44] Tinu Varghese, R. Sheela Kumari, P. S. Mathuranath, and N. Albert Singh. Performance evaluation of bacterial foraging optimization algorithm for the early diagnosis and tracking of alzheimer's disease. In *Swarm, Evolutionary, and Memetic Computing*, pages 41–48, Berlin, Heidelberg, 2012 (Accessed July 15th, 2019). Springer Berlin Heidelberg.

[45] Xin-She Yang. *Nature-Inspired Metaheuristic Algorithms*. Luniver Press, 2008 (Accessed July 22nd, 2019).

[46] Xin-She Yang. A new metaheuristic bat-inspired algorithm. *arXiv*, 1004.4170, 2010 (Accessed July 10th, 2019).

[47] Chunkai Zhang and Huihe Shao. An ann's evolved by a new evolutionary system and its application. *Proceedings of the IEEE Conference on Decision and Control*, 4, December 2000 (Accessed July 5th, 2019).

Appendix A

Additional Experimental Analysis

A.1 Parameters

Dataset	W	C1	C2
Auto	0.83705	1.06087	1.81788
BreastCancer	0.61932	1.45417	1.32986
ComputerHardware	0.46905	1.76786	1.25082
Facebook	0.82680	1.34440	1.27918
Forest Fires	0.72215	1.33069	1.38188
Glass	0.73130	1.35746	1.04479
Ionosphere	0.58679	1.69276	1.43742
Iris	0.48413	1.08102	2.00000
LandCover	0.75536	1.20004	1.26056
MNIST	0.82183	1.01534	1.00000
Music	0.51093	1.90498	2.00000
Musk	0.64268	1.63367	1.65435
Parkinsons	0.80844	1.24275	1.38543
Residential	0.47863	1.60751	1.56877
Scadi	0.41964	1.98139	1.55482
Seisures	0.71530	2.00000	1.20693
Servo	0.71271	1.83845	1.27641
Sonar	0.40000	1.00000	2.00000
Soybean	0.58481	2.00000	1.84532
Spambase	0.62720	1.06945	1.83581
Thyroid	0.73621	2.00000	1.37548
Wine	0.50136	1.71742	1.63091
Zoo	0.56533	1.26656	1.69603

Table A.1: PSO with Sigmoid Activation function

Dataset	W	C1	C2
Auto	0.46646	1.57942	1.86741
BreastCancer	0.81480	1.57010	1.72631
ComputerHardware	0.73637	1.90832	1.65488
Facebook	0.46242	1.36967	1.93937
ForestFires	0.65011	1.69445	1.00000
Glass	0.63624	1.28641	1.36545
Ionosphere	0.40000	2.00000	1.55997
Iris	0.61732	1.69883	1.48613
LandCover	0.68170	1.47533	1.25672
MNIST	0.57713	1.08182	1.91873
Music	0.65938	1.27090	1.77878
Musk	0.55002	1.89787	1.73336
Parkinson's	0.78695	1.20844	1.40495
Residential	0.90000	1.04329	1.73309
Scadi	0.64163	1.53760	1.94999
Seisures	0.40000	2.00000	2.00000
Servo	0.42106	1.75555	1.65312
Sonar	0.57930	1.80493	1.20578
Soybean	0.40000	2.00000	2.00000
Spambase	0.46368	1.76021	1.91668
Thyroid	0.40000	1.97971	2.00000
Wine	0.40372	1.81842	1.89869
Zoo	0.56008	1.62019	2.00000

Table A.2: PSO with Elliot Activation function

Dataset	W	C1	C2
Auto	0.70219	1.51438	1.40868
BreastCancer	0.63404	1.32337	1.60942
ComputerHardware	0.76408	1.12287	1.28198
Facebook	0.66595	1.41551	1.46327
ForestFires	0.52755	2.00000	1.55963
Glass	0.80036	1.19798	1.88671
Ionosphere	0.40000	1.09776	2.00000
Iris	0.86177	1.57419	1.26051
LandCover	0.43740	1.88197	1.74263
MNIST	0.79201	1.47311	1.12240
Music	0.81467	1.70113	1.18764
Musk	0.62141	2.00000	1.52753
Parkinsons	0.40000	1.83217	1.21257
Residential	0.89346	1.81393	1.22242
Scadi	0.41936	1.03618	1.96767
Seisures	0.47080	1.75507	2.00000
Servo	0.88286	1.92319	1.59838
Sonar	0.66143	1.27394	1.64789
Soybean	0.55359	1.75064	1.79964
Spambase	0.59274	1.91509	1.66358
Thyroid	0.50370	1.86830	1.86269
Wine	0.78485	1.48092	1.00000
Zoo	0.45343	1.55667	1.60242

Table A.3: PSO with Lecun Activation function

Dataset	Employed Percentage
Auto	0.54123
BreastCancer	0.50000
ComputerHardware	0.50697
Facebook	0.61422
ForestFires	0.51563
Glass	0.57400
Ionosphere	0.62019
Iris	0.94317
LandCover	0.58216
MNIST	0.63925
Music	0.60798
Musk	0.60318
Parkinsons	0.66325
Residential	0.64061
Scadi	0.89937
Seisures	0.56041
Servo	0.82494
Sonar	0.82849
Soybean	0.61406
Spambase	0.60198
Thyroid	0.61442
Wine	0.56693
Zoo	0.78172

Table A.4: ABC with Tanh Activation function

Auto	0.78660
BreastCancer	0.54536
ComputerHardware	0.89980
Facebook	0.63126
ForestFires	0.51487
Glass	0.62034
Ionosphere	0.71070
Iris	0.50663
LandCover	0.61945
MNIST	0.60582
Music	0.56675
Musk	0.63379
Parkinsons	0.54831
Residential	0.63452
Scadi	0.66095
Seisures	0.88590
Servo	0.50126
Sonar	0.60714
Soybean	0.58766
Spambase	0.64929
Thyroid	0.52667
Wine	0.63871
Zoo	0.89995

Table A.5: ABC with Sigmoid Activation function

Dataset	Employed Percentage
Auto	0.50000
BreastCancer	0.50439
ComputerHardware	0.50502
Facebook	0.58878
ForestFires	0.50038
Glass	0.58195
Ionosphere	0.64956
Iris	0.50200
LandCover	0.58910
MNIST	0.65101
Music	0.66758
Musk	0.67177
Parkinsons	0.74989
Residential	0.60832
Scadi	0.88662
Seisures	0.63516
Servo	0.50000
Sonar	0.89996
Soybean	0.60898
Spambase	0.75471
Thyroid	0.50000
Wine	0.66985
Zoo	0.67973

Table A.6: ABC with Elliot Activation function

Dataset	Employed Percentage
Auto	0.56615
BreastCancer	0.50000
ComputerHardware	0.57656
Facebook	0.60601
ForestFires	0.58989
Glass	0.72011
Ionosphere	0.50008
Iris	0.57045
LandCover	0.65168
MNIST	0.63548
Music	0.62865
Musk	0.57199
Parkinsons	0.65603
Residential	0.59640
Scadi	0.64257
Seisures	0.57173
Servo	0.50000
Sonar	0.51640
Soybean	0.57647
Spambase	0.54354
Thyroid	0.74567
Wine	0.58695
Zoo	0.61441

Table A.7: ABC with Lecun Activation function

Dataset	Q_{test}	Pheromone Evaporation Rate
Auto	0.39153	0.30338
BreastCancer	0.48543	0.59716
ComputerHardware	0.33917	0.02488
Facebook	0.36361	0.84594
ForestFires	0.14425	0.48530
Glass	0.34860	0.75830
Ionosphere	0.11047	0.01379
Iris	0.50000	0.58932
LandCover	0.48083	0.30098
MNIST	0.17440	0.54300
Music	0.02528	0.78345
Musk	0.50000	0.90000
Parkinsons	0.37201	0.13033
Residential	0.01000	0.80969
Scadi	0.01119	0.09019
Seisures	0.50000	0.90000
Servo	0.09064	0.62956
Sonar	0.29345	0.01145
Soybean	0.01000	0.90000
Spambase	0.50000	0.90000
Thyroid	0.28933	0.13749
Wine	0.49979	0.01781
Zoo	0.18266	0.73496

Table A.8: ACO with Tanh Activation function parameters

Dataset	Q_{test}	Pheromone Evaporation Rate
Auto	0.02862	0.38021
BreastCancer	0.27591	0.62868
ComputerHardware	0.32483	0.77930
Facebook	0.22423	0.66595
ForestFires	0.15780	0.10185
Glass	0.38678	0.61407
Ionosphere	0.50000	0.78783
Iris	0.42727	0.90000
LandCover	0.50000	0.65665
MNIST	0.50000	0.90000
Music	0.46529	0.89949
Musk	0.50000	0.90000
Parkinsons	0.36095	0.16790
Residential	0.50000	0.90000
Scadi	0.25741	0.50503
Seisures	0.18229	0.59227
Servo	0.04662	0.14037
Sonar	0.49014	0.87579
Soybean	0.50000	0.62784
Spambase	0.50000	0.90000
Thyroid	0.47868	0.30177
Wine	0.50000	0.54757
Zoo	0.50000	0.63882

Table A.9: ACO with Sigmoid Activation function Parameters

Dataset	Q_{test}	Pheromone Evaporation Rate
Auto	0.38405	0.26318
BreastCancer	0.40414	0.51630
ComputerHardware	0.26133	0.44897
Facebook	0.38605	0.88951
ForestFires	0.11183	0.71027
Glass	0.50000	0.60416
Ionosphere	0.36550	0.33793
Iris	0.50000	0.90000
LandCover	0.14909	0.29715
MNIST	0.15333	0.69392
Music	0.12462	0.66338
Musk	0.34182	0.33412
Parkinsons	0.24656	0.77270
Residential	0.46306	0.62000
Scadi	0.21279	0.60626
Seisures	0.50000	0.90000
Servo	0.03574	0.01534
Sonar	0.44540	0.23261
Soybean	0.50000	0.90000
Spambase	0.22700	0.63271
Thyroid	0.26377	0.15429
Wine	0.12445	0.73191
Zoo	0.37010	0.30049

Table A.10: ACO with Elliot Activation function parameters

Dataset	Q_{test}	Pheromone Evaporation Rate
Auto	0.33104	0.01932
BreastCancer	0.05487	0.56010
ComputerHardware	0.32286	0.42726
Facebook	0.36686	0.87271
ForestFires	0.18196	0.83862
Glass	0.25897	0.57755
Ionosphere	0.01000	0.01000
Iris	0.38785	0.31808
LandCover	0.45810	0.71182
MNIST	0.06030	0.65644
Music	0.09263	0.04953
Musk	0.50000	0.29454
Parkinsons	0.16854	0.42859
Residential	0.50000	0.24646
Scadi	0.01160	0.04608
Seisures	0.01000	0.82674
Servo	0.28052	0.05259
Sonar	0.01749	0.20750
Soybean	0.42790	0.90000
Spambase	0.23994	0.67601
Thyroid	0.49585	0.40292
Wine	0.15846	0.29466
Zoo	0.43632	0.27330

Table A.11: ACO with Lecun Activation function parameters

Dataset	$r0$	$v0$	F_{max}	α & γ
Auto	0.34485	1.00000	0.24897	0.87427
BreastCancer	0.06311	0.68865	0.24984	0.18979
ComputerHardware	0.00100	1.00000	1.00000	1.00000
Facebook	0.00100	0.44698	0.00100	1.00000
ForestFires	0.95247	0.51534	0.78999	0.53365
Glass	0.50558	0.49256	0.79189	0.89989
Ionosphere	0.48505	0.19570	0.27871	0.87616
Iris	0.14896	1.00000	1.00000	1.00000
LandCover	0.30430	0.08469	0.12651	0.95270
MNIST	0.04754	0.69699	0.54300	0.91771
Music	0.55045	0.51341	0.00100	1.00000
Musk	0.32916	0.21978	0.19112	0.90193
Parkinsons	0.62636	0.43727	0.15052	0.65594
Residential	0.97721	0.82290	0.98730	0.98755
Scadi	0.61222	0.98387	0.51685	0.36239
Seisures	0.13025	0.65800	0.45645	0.86621
Servo	0.70827	0.55074	0.69523	0.69411
Sonar	0.22827	0.16836	0.21194	0.80964
Soybean	0.52383	0.31926	0.76684	1.00000
Spambase	0.29235	0.67015	0.27585	0.92003
Thyroid	1.00000	0.79276	0.16148	0.54788
Wine	0.70727	0.22318	0.25992	0.89039
Zoo	0.12090	0.87464	0.82019	0.62183

Table A.12: BA with Tanh Activation function parameters

Dataset	$r0$	$v0$	F_{max}	α & γ
Auto	0.58465	0.63426	0.62016	0.95227
BreastCancer	0.32448	0.23282	0.37330	0.45278
ComputerHardware	0.00100	1.00000	0.72998	1.00000
Facebook	0.00100	1.00000	1.00000	1.00000
ForestFires	0.19608	0.79123	0.00100	0.76037
Glass	0.17420	0.67727	0.67265	0.89275
Ionosphere	0.62121	0.28637	0.54440	0.94273
Iris	0.76211	0.39552	0.00860	0.84746
LandCover	0.37119	0.24422	0.08750	0.87695
MNIST	0.97475	0.56716	0.13984	0.78007
Music	0.17051	0.79107	0.86296	0.89380
Musk	0.54337	0.12445	0.63993	1.00000
Parkinsons	0.37060	0.02529	0.66420	0.97571
Residential	0.00100	1.00000	0.00100	1.00000
Scadi	0.10276	0.27572	0.51610	0.82290
Seisures	0.42790	0.58129	0.57584	0.85128
Servo	0.00100	0.61555	0.47612	0.86130
Sonar	0.10130	0.29543	0.23294	0.87916
Soybean	0.00100	0.20311	0.47971	1.00000
Spambase	0.56936	0.85510	0.31951	0.86627
Thyroid	0.07505	0.31311	0.72780	0.95885
Wine	0.91661	1.00000	0.49208	0.65158
Zoo	0.40363	0.33902	0.71544	0.98705

Table A.13: BA with Sigmoid Activation function parameters

Dataset	$r0$	$v0$	F_{max}	α & γ
Auto	0.23549	0.72139	0.17815	0.78568
BreastCancer	0.75860	0.53609	0.22111	0.57505
ComputerHardware	0.13180	0.71490	0.58597	1.00000
Facebook	0.02928	0.97713	0.04238	0.94512
ForestFires	0.33310	0.14543	0.92530	0.73558
Glass	0.67916	0.27805	0.52715	0.93270
Ionosphere	0.67449	0.43591	0.40648	0.83294
Iris	0.55598	0.60556	0.13416	0.62458
LandCover	0.84904	0.24468	0.08921	0.65873
MNIST	0.28019	0.90847	0.53502	0.80145
Music	0.00100	0.89344	1.00000	1.00000
Musk	0.28993	0.11304	0.79694	0.97632
Parkinsons	0.00100	0.31488	0.28055	0.39867
Residential	0.89159	0.01253	0.30639	0.21396
Scadi	0.95353	0.82739	0.32497	0.85926
Seisures	0.00100	0.08091	0.00100	1.00000
Servo	0.11859	0.99145	0.01200	0.86555
Sonar	0.70112	0.42440	0.38514	0.14888
Soybean	0.39555	1.00000	0.30420	0.96294
Spambase	0.46724	0.15285	0.44888	0.94314
Thyroid	0.45205	0.81166	0.56905	0.92351
Wine	0.61920	0.38546	0.84548	0.81301
Zoo	0.47676	0.70729	0.74511	0.71892

Table A.14: BA with Elliot Activation function parameters

Dataset	$r0$	$v0$	F_{max}	α & γ
Auto	0.44949	1.00000	0.00100	1.00000
BreastCancer	0.92683	0.23203	0.54539	0.63434
ComputerHardware	0.48233	0.46005	0.00100	1.00000
Facebook	0.90256	0.19181	0.87542	0.67074
ForestFires	0.85109	0.29560	0.32069	0.54836
Glass	1.00000	0.61272	0.75871	0.89857
Ionosphere	0.96371	0.41234	0.83101	0.64036
Iris	0.23458	0.68540	0.70426	0.59742
LandCover	0.74072	0.00100	0.24978	0.98623
MNIST	0.78735	0.00100	0.62629	1.00000
Music	0.77364	0.50662	0.09017	0.92441
Musk	1.00000	0.24311	0.42691	0.92039
Parkinsons	0.14317	1.00000	0.66702	0.83814
Residential	0.01857	0.32788	0.22649	0.68122
Scadi	0.80779	0.24929	0.69082	0.53008
Seisures	0.36283	0.74493	0.42753	0.87612
Servo	0.44874	0.43382	0.74585	0.86018
Sonar	0.69757	0.27246	0.12037	0.77421
Soybean	0.39836	0.15062	0.25859	1.00000
Spambase	0.72291	0.41559	0.80329	0.81138
Thyroid	0.71292	0.66990	0.38368	0.93939
Wine	0.56455	0.65883	0.15930	0.55413
Zoo	0.00100	0.92456	1.00000	1.00000

Table A.15: BA with Lecun Activation function parameters

Dataset	N_{ed}	N_{rs}	N_c	SL	$StepSize$	D_{attr}	W_{attr}	H_{rep}	W_{rep}	P_{ed}
Auto	1	5	1	4	1.0000	0.9910	0.9900	0.0100	0.9900	0.0100
BreastCancer	4	4	5	8	0.3781	0.0645	0.2987	0.8339	0.2549	0.0971
CompHardware	10	4	1	6	1.0000	0.0100	0.9900	0.9900	0.0100	0.0100
Facebook	10	5	1	4	1.0000	0.9910	0.8974	0.0100	0.0100	0.0100
ForestFires	2	3	5	1	0.1426	0.4745	0.9679	0.2160	0.4215	0.3451
Glass	8	1	3	9	0.0306	0.0100	0.6078	0.2281	0.9024	0.0387
Ionosphere	5	4	3	10	0.2620	0.3738	0.5489	0.3935	0.8622	0.3208
Iris	1	3	1	7	0.4591	0.2180	0.3676	0.6725	0.9090	0.8019
LandCover	4	1	3	10	0.2144	0.9910	0.0100	0.9900	0.0100	0.3017
MNIST	7	1	3	2	0.0100	0.0100	0.0370	0.9417	0.6542	0.4046
Music	10	5	1	10	1.0000	0.0100	0.9900	0.0100	0.9900	0.0100
Musk	6	1	1	8	0.0100	0.0100	0.9900	0.0100	0.9900	0.0100
Parkinsons	8	4	1	7	0.2249	0.3737	0.2594	0.4019	0.1359	0.9176
Residential	8	5	1	8	1.0000	0.2209	0.0100	0.0100	0.9900	0.0100
Scadi	1	4	2	1	0.6760	0.4231	0.9410	0.0181	0.6385	0.7221
Seisures	7	3	1	4	0.1752	0.1359	0.9601	0.1496	0.8113	0.5521
Servo	10	1	5	5	0.0100	0.9910	0.0100	0.9900	0.0100	0.0100
Sonar	10	4	5	6	0.7127	0.8120	0.2226	0.3945	0.4788	0.0837
Soybean	10	5	5	7	0.0100	0.0100	0.0100	0.0100	0.9900	0.0100
Spambase	6	5	3	10	0.0100	0.9910	0.0100	0.9900	0.9900	0.99
Thyroid	1	2	1	3	0.6946	0.6766	0.7222	0.6773	0.4195	0.7105
Wine	7	3	1	9	0.1229	0.6086	0.6935	0.7535	0.6710	0.4971
Zoo	4	4	1	10	0.2067	0.1350	0.1852	0.5602	0.9651	0.0866

Table A.16: BFA with Tanh Activation function parameters

Dataset	N_{ed}	N_{rs}	N_c	SL	$StepSize$	D_{attr}	W_{attr}	H_{rep}	W_{rep}	P_{ed}
Auto	8	5	4	4	0.7783	0.6588	0.6068	0.8317	0.6154	0.4580
BreastCancer	3	5	5	10	1	0.01	0.99	0.01	0.99	0.01
CompHardware	4	1	2	5	0.7139	0.1145	0.8197	0.1000	0.8340	0.1936
Facebook	1	5	3	4	0.01	0.01	0.01	0.99	0.01	0.99
ForestFires	10	1	4	6	0.01	0.0102	0.01	0.8874	0.99	0.01
Glass	1	5	1	7	1	0.01	0.01	0.01	0.99	0.99
Ionosphere	5	4	1	4	0.1609	0.5932	0.01	0.01	0.01	0.4969
Iris	9	3	3	9	0.7155	0.7289	0.8324	0.9900	0.8195	0.8233
LandCover	1	5	5	10	1	0.01	0.01	0.01	0.99	0.01
MNIST	1	5	1	3	0.4822	0.5289	0.9713	0.7361	0.3818	0.5312
Music	5	5	4	1	0.5039	0.5975	0.4588	0.5424	0.6875	0.0507
Musk	1	2	5	5	1	0.991	0.01	0.99	0.01	0.99
Parkinsons	1	1	2	7	1	0.991	0.01	0.99	0.99	0.99
Residential	10	2	3	1	1	0.991	0.01	0.01	0.99	0.99
Scadi	1	3	1	4	1	0.991	0.2376	0.99	0.99	0.99
Seisures	1	5	1	1	1	0.991	0.01	0.99	0.01	0.01
Servo	1	5	5	1	1	0.991	0.01	0.99	0.99	0.01
Sonar	1	1	1	10	1	0.01	0.99	0.01	0.99	0.99
Soybean	1	5	5	8	1	0.01	0.01	0.01	0.3192	0.01
Spambase	7	4	3	4	0.2533	0.4553	0.5356	0.7053	0.8894	0.8994
Thyroid	8	5	1	1	1	0.991	0.01	0.99	0.01	0.99
Wine	8	1	1	5	1	0.01	0.99	0.01	0.7869	0.99
Zoo	3	5	4	7	0.8872	0.2420	0.6520	0.3066	0.3232	0.8435

Table A.17: BFA with Sigmoid Activation function parameters

Dataset	N_{ed}	N_{rs}	N_c	SL	$StepSize$	D_{attr}	W_{attr}	H_{rep}	W_{rep}	P_{ed}
BreastCancer	5	5	1	1	0.7007	0.1534	0.0904	0.2524	0.8956	0.6822
CompHardware	6	5	5	8	1	0.01	0.01	0.01	0.6867	0.01
Facebook	5	5	1	10	1	0.991	0.99	0.99	0.01	0.01
ForestFires	1	5	3	1	1	0.9907	0.99	0.4538	0.9714	0.1623
Glass	7	1	3	3	0.1423	0.3680	0.3386	0.3499	0.9471	0.3464
Ionosphere	5	4	5	7	0.1743	0.9296	0.2348	0.8410	0.5577	0.8129
Iris	10	3	2	10	1	0.01	0.0696	0.0983	0.9757	0.01
LandCover	1	5	5	2	1	0.0775	0.2572	0.99	0.7622	0.9478
MNIST	2	1	2	1	0.5525	0.3898	0.9375	0.0853	0.4414	0.5496
Music	10	2	1	3	1	0.01	0.99	0.99	0.01	0.01
Musk	9	2	1	5	0.0229	0.2293	0.0780	0.8628	0.4407	0.1573
Parkinsons	9	2	4	10	0.0697	0.7274	0.0415	0.6457	0.1426	0.7317
Residential	5	2	2	6	0.8931	0.5777	0.1115	0.4142	0.99	0.4127
Scadi	5	3	3	10	0.6217	0.3000	0.8130	0.2869	0.9813	0.6726
Seisures	10	3	1	8	0.01	0.01	0.01	0.01	0.01	0.01
Servo	9	5	1	10	0.01	0.01	0.01	0.8216	0.99	0.01
Sonar	10	5	3	6	0.01	0.4720	0.5888	0.8291	0.2881	0.0120
Soybean	1	5	3	6	0.3321	0.6700	0.2079	0.6881	0.3379	0.6694
Spambase	8	3	4	5	0.6580	0.4646	0.1393	0.4835	0.5466	0.0190
Thyroid	10	5	3	1	0.0848	0.3550	0.5208	0.4641	0.3889	0.0193
Wine	2	4	1	10	0.8241	0.0133	0.4190	0.01	0.4081	0.5382
Zoo	7	4	2	10	0.2327	0.2618	0.5677	0.6176	0.3124	0.6742

Table A.18: BFA with Elliot Activation function parameters

Dataset	N_{ed}	N_{rs}	N_c	SL	$StepSize$	D_{attr}	W_{attr}	H_{rep}	W_{rep}	P_{ed}
Auto	3	4	1	1	0.1761	0.2780	0.4595	0.9368	0.1948	0.2098
BreastCancer	4	1	1	6	0.1340	0.1711	0.4718	0.4530	0.7695	0.0100
CompHardware	10	4	1	5	0.1346	0.7023	0.7503	0.4691	0.7335	0.2859
Facebook	9	5	1	6	1.0000	0.9910	0.0100	0.0100	0.0100	0.0100
ForestFires	1	4	3	9	0.2673	0.6325	0.2511	0.7335	0.8503	0.0408
Glass	1	5	1	10	0.0100	0.0100	0.0100	0.9900	0.0100	0.99
Ionosphere	9	3	4	4	0.3922	0.0119	0.7509	0.0239	0.0540	0.5155
Iris	6	3	1	2	0.5826	0.0100	0.9900	0.4919	0.3699	0.01
LandCover	3	4	4	6	0.1144	0.8599	0.4696	0.7784	0.1377	0.3289
MNIST	7	5	5	3	0.4362	0.5910	0.4640	0.5782	0.5422	0.3085
Music	6	3	5	9	0.2548	0.0240	0.2675	0.1381	0.3030	0.1063
Musk	9	5	1	1	0.0216	0.9910	0.0100	0.9900	0.0100	0.3962
Parkinsons	1	4	1	9	0.3477	0.5401	0.4116	0.4973	0.3740	0.3677
Residential	10	3	2	8	1.0000	0.8043	0.0868	0.3804	0.0100	0.0100
Scadi	6	5	4	4	0.3225	0.3878	0.3554	0.4631	0.2039	0.7891
Seisures	2	2	2	1	0.4258	0.4082	0.4009	0.3913	0.5343	0.3778
Servo	8	4	5	4	0.0375	0.0667	0.0794	0.0597	0.8105	0.1933
Sonar	9	3	2	10	0.2976	0.4422	0.9900	0.4763	0.6218	0.7771
Soybean	10	5	2	6	0.0544	0.4991	0.6009	0.4872	0.7065	0.4850
Spambase	3	1	1	3	0.3156	0.2694	0.1331	0.3252	0.9527	0.7544
Thyroid	10	4	1	10	0.01	0.01	0.01	0.01	0.99	0.01
Wine	1	2	4	9	0.3437	0.2142	0.2365	0.3609	0.3663	0.3623
Zoo	10	1	1	7	1	0.01	0.01	0.01	0.3833	0.01

Table A.19: BFA with Lecun Activation function parameters

Dataset	$Step_{ind}Initial$	$Step_{vol}Initial$	$Step_{ind}Final$	$Step_{vol}Final$
Auto	0.0640326500	0.0611531800	0.0037432700	0.0040750600
BreastCancer	0.0997775700	0.0025955700	0.0066656700	0.0002719500
ComputerHardware	0.0070230000	0.0729206000	0.0062393900	0.0007269800
Facebook	0.0153449800	0.0565784600	0.0008439000	0.0040793600
ForestFires	0.0183953200	0.0231112200	0.0029121400	0.0047482100
Glass	0.0575250700	0.0294339700	0.0100000000	0.0100000000
Ionosphere	0.0293133900	0.0083447100	0.0097576500	0.0045782400
Iris	0.0225809700	0.0489447500	0.0100000000	0.0100000000
LandCover	0.1000000000	0.0721622113	0.0000010000	0.0100000000
MNIST	0.0099968900	0.0067343000	0.0093176600	0.0038863300
Music	0.0922914200	0.0208128300	0.0067927100	0.0035945100
Musk	0.0607095131	0.0230584967	0.0100000000	0.0000010000
Parkinsons	0.1000000000	0.0911862197	0.0100000000	0.0000010000
Residential	0.0768214425	0.0828153911	0.0000010000	0.0000010000
Scadi	0.0990675800	0.0935163000	0.0030814300	0.0093013100
Seisures	0.1000000000	0.0000100000	0.0100000000	0.0000010000
Servo	0.0363895100	0.0591016900	0.0079702600	0.0036692400
Sonar	0.0063908761	0.0072706471	0.0051710154	0.0049251287
Soybean	0.0892147800	0.0030584900	0.0100000000	0.0027920300
Spambase	0.1000000000	0.0000100000	0.0100000000	0.0000010000
Thyroid	0.1000000000	0.0267287400	0.0100000000	0.0100000000
Wine	0.0953838500	0.0995659400	0.0081501100	0.0067862900
Zoo	0.0994644200	0.0652181700	0.0100000000	0.0100000000

Table A.20: FSS with Tanh Activation function parameters

Dataset	$Step_{ind}Initial$	$Step_{vol}Initial$	$Step_{ind}Final$	$Step_{vol}Final$
Auto	0.0865083500	0.0855141600	0.0075388300	0.0012223900
BreastCancer	0.0694844274	0.0828942023	0.0100000000	0.0000001000
ComputerHardware	0.1000000000	0.0781649300	0.0100000000	0.0100000000
Facebook	0.1000000000	0.1000000000	0.0100000000	0.0100000000
ForestFires	0.0189834500	0.0091454100	0.0100000000	0.0031623100
Glass	0.1000000000	0.0382578500	0.0100000000	0.0100000000
Ionosphere	0.1000000000	0.0513173800	0.0100000000	0.0100000000
Iris	0.1000000000	0.0676341900	0.0100000000	0.0100000000
LandCover	0.0387744900	0.0067974700	0.0082190300	0.0039586900
MNIST	0.1000000000	0.0000100000	0.0100000000	0.0000010000
Music	0.0932727000	0.0706669000	0.0012215800	0.0034138500
Musk	0.1000000000	0.0000100000	0.0100000000	0.0000010000
Parkinsons	0.1000000000	0.0644151293	0.0100000000	0.0000010000
Residential	0.0914428281	0.0688090166	0.0100000000	0.0000010000
Scadi	0.0879453000	0.0885157000	0.0044284600	0.0094707100
Seisures	0.1000000000	0.0679101161	0.0100000000	0.0000001000
Servo	0.0602600443	0.0796983812	0.0100000000	0.0000010000
Sonar	0.0435160300	0.0669797000	0.0035184100	0.0006548600
Soybean	0.1000000000	0.0260146100	0.0100000000	0.0100000000
Spambase	0.1000000000	0.0000100000	0.0100000000	0.0000001000
Thyroid	0.1000000000	0.0627171349	0.0100000000	0.0000010000
Wine	0.0996550500	0.0006072000	0.0089462200	0.0001874700
Zoo	0.1000000000	0.0220724952	0.0100000000	0.0000010000

Table A.21: FSS with Sigmoid Activation function parameters

Dataset	$Step_{ind}Initial$	$Step_{vol}Initial$	$Step_{ind}Final$	$Step_{vol}Final$
Auto	0.0202738400	0.0808033300	0.0011837000	0.0023883800
BreastCancer	0.0918827437	0.1000000000	0.0100000000	0.0000001000
ComputerHardware	0.0345840100	0.0463870500	0.0015548200	0.0076778500
Facebook	0.0455971600	0.0538052500	0.0053228500	0.0061286000
ForestFires	0.0565202200	0.0364211200	0.0058144200	0.0086454900
Glass	0.0824177800	0.0430490100	0.0100000000	0.0100000000
Ionosphere	0.0953786700	0.0206962900	0.0055805700	0.0086488300
Iris	0.1000000000	0.1000000000	0.0100000000	0.0000010000
LandCover	0.0928146316	0.0117342667	0.0100000000	0.0000010000
MNIST	0.0742384798	0.0971500699	0.0000010000	0.0000010000
Music	0.0831942700	0.0173658000	0.0039808500	0.0067064500
Musk	0.0710079894	0.1000000000	0.0100000000	0.0000010000
Parkinsons	0.0908172300	0.0518789500	0.0034499300	0.0061594900
Residential	0.0282791540	0.0022426659	0.0100000000	0.0000010000
Scadi	0.1000000000	0.0000100000	0.0100000000	0.0000010000
Seisures	0.1000000000	0.0000100000	0.0100000000	0.0000001000
Servo	0.0462913000	0.0387172300	0.0034898100	0.0055840300
Sonar	0.0986231000	0.0234594000	0.0075269900	0.0072530900
Soybean	0.1000000000	0.0617075787	0.0100000000	0.0000010000
Spambase	0.0889792416	0.0103953581	0.0100000000	0.0000001000
Thyroid	0.1000000000	0.0473356900	0.0100000000	0.0100000000
Wine	0.0529812100	0.0992855500	0.0097362600	0.0094952700
Zoo	0.0939397800	0.0125134000	0.0079166400	0.0048506700

Table A.22: FSS with Elliot Activation function parameters

Dataset	$Step_{ind}Initial$	$Step_{vol}Initial$	$Step_{ind}Final$	$Step_{vol}Final$
Auto	0.0513084900	0.0634048900	0.0013401600	0.0062200600
BreastCancer	0.0364948900	0.0923910600	0.0067421200	0.0098851000
ComputerHardware	0.0396408500	0.0410398400	0.0087462500	0.0041153700
Facebook	0.0137287200	0.0830123600	0.0007287100	0.0071122800
ForestFires	0.0683098200	0.0147067900	0.0098141300	0.0011630600
Glass	0.1000000000	0.0000100000	0.0000010000	0.0000010000
Ionosphere	0.0395329000	0.0907069200	0.0098919800	0.0006769800
Iris	0.0393688107	0.0174293139	0.0100000000	0.0000010000
LandCover	0.1000000000	0.0584881300	0.0087860700	0.0100000000
MNIST	0.0894926221	0.0243283858	0.0100000000	0.0000010000
Music	0.0949245300	0.0477674200	0.0022515400	0.0015795500
Musk	0.0850766856	0.0000100000	0.0100000000	0.0000010000
Parkinsons	0.0972836200	0.0017806600	0.0081037700	0.0008075620
Residential	0.0754420274	0.0037642296	0.0100000000	0.0000010000
Scadi	0.0991158100	0.0066083000	0.0019490800	0.0006552300
Seisures	0.1000000000	0.0000100000	0.0100000000	0.0000001000
Servo	0.0766621100	0.0847049000	0.0049256200	0.0012807200
Sonar	0.0807767800	0.0918918500	0.0037735900	0.0070203900
Soybean	0.0830319954	0.0138042117	0.0100000000	0.0000010000
Spambase	0.0820187557	0.0681210905	0.0000010000	0.0100000000
Thyroid	0.1000000000	0.0211667000	0.0100000000	0.0100000000
Wine	0.0910856542	0.0000100000	0.0100000000	0.0000010000
Zoo	0.0716685600	0.0108639400	0.0060267400	0.0100000000

Table A.23: FSS with Lecun Activation function parameters

Dataset	α	$Beta_{min}$	γ
Auto	0.69796	0.89041	8
BreastCancer	0.68768	0.12163	26
ComputerHardware	0.05675	0.34173	3
Facebook	0.35405	0.11665	10
ForestFires	0.01000	0.71873	1
Glass	0.01000	0.01000	20
Ionosphere	0.19355	0.01000	2
Iris	0.99000	0.01000	17
LandCover	0.01000	0.03488	27
MNIST	0.07679	0.28114	29
Music	0.25288	0.02007	26
Musk	0.06102	0.11413	1
Parkinsons	0.40974	0.41740	25
Residential	0.83574	0.28521	25
Scadi	0.32136	0.97424	6
Seisures	0.11818	0.06700	25
Servo	0.01000	0.99000	17
Sonar	0.16235	0.16790	11
Soybean	0.25374	0.02628	11
Spambase	0.03018	0.25528	16
Thyroid	0.84762	0.27943	23
Wine	0.49612	0.42815	20
Zoo	0.57417	0.99000	26

Table A.24: Firefly with Tanh Activation function parameters

Dataset	α	$Beta_{min}$	γ
Auto	0.78181	0.67543	4
BreastCancer	0.25441	0.78526	5
ComputerHardware	0.44219	0.15389	28
Facebook	0.55747	0.69859	9
ForestFires	0.60150	0.41103	4
Glass	0.74695	0.52468	28
Ionosphere	0.64915	0.46333	23
Iris	0.34526	0.01000	4
LandCover	0.51456	0.37104	3
MNIST	0.66148	0.04160	4
Music	0.63216	0.36638	29
Musk	0.56139	0.20703	14
Parkinsons	0.79498	0.90960	24
Residential	0.89561	0.80315	28
Scadi	0.01000	0.99000	9
Seisures	0.91485	0.19990	12
Servo	0.95991	0.78515	10
Sonar	0.08485	0.49518	24
Soybean	0.57042	0.24405	23
Spambase	0.53521	0.18876	6
Thyroid	0.76920	0.18342	24
Wine	0.33759	0.01000	4
Zoo	0.66725	0.18199	2

Table A.25: Firefly with Sigmoid Activation function parameters

Dataset	α	$Beta_{min}$	γ
Auto	0.50858	0.43192	21
BreastCancer	0.21874	0.78227	22
ComputerHardware	0.81665	0.78647	1
Facebook	0.94696	0.32634	1
ForestFires	0.01000	0.99000	8
Glass	0.28747	0.07476	11
Ionosphere	0.01000	0.49375	13
Iris	0.47845	0.18778	5
LandCover	0.04285	0.98321	1
MNIST	0.05159	0.20044	9
Music	0.53145	0.84134	5
Musk	0.27779	0.14779	3
Parkinsons	0.16373	0.94632	24
Residential	0.38722	0.16757	1
Scadi	0.17342	0.98174	13
Seisures	0.14321	0.11455	14
Servo	0.52594	0.61814	23
Sonar	0.08731	0.68530	15
Soybean	0.23149	0.12717	11
Spambase	0.21200	0.34342	26
Thyroid	0.29090	0.26378	17
Wine	0.70264	0.84819	8
Zoo	0.64735	0.79196	20

Table A.26: Firefly with Elliot Activation function parameters

Dataset	α	$Beta_{min}$	γ
Auto	0.97906	0.56663	8
BreastCancer	0.81637	0.84690	21
ComputerHardware	0.64632	0.84429	6
Facebook	0.79580	0.41168	29
ForestFires	0.01000	0.01000	21
Glass	0.09709	0.18223	14
Ionosphere	0.12749	0.88095	16
Iris	0.70536	0.07223	7
LandCover	0.01000	0.01000	13
MNIST	0.02816	0.07311	1
Music	0.14871	0.09838	23
Musk	0.03127	0.03789	30
Parkinsons	0.99000	0.99000	1
Residential	0.88897	0.41980	16
Scadi	0.01000	0.01000	20
Seisures	0.01000	0.01000	1
Servo	0.39368	0.17034	5
Sonar	0.01000	0.01000	25
Soybean	0.05407	0.10811	12
Spambase	0.04250	0.16333	18
Thyroid	0.09265	0.26077	5
Wine	0.06866	0.36236	24
Zoo	0.77611	0.45906	30

Table A.27: Firefly with Lecun Activation function parameters

A.2 Average and Standard Deviation of Results

	Average	StdDev	Average	StdDev
TrainingLoss	PSO	PSO	ABC	ABC
Auto	12.54804977	2.442308972	32.12542668	18.08741216
BreastCancer	0	0	0	0
ComputerHardware	5391.6105	7486.412	20766.024	9937.942
Facebook	3.511749695	0.581738784	41.41198953	67.2973
ForestFires	3.511749695	0.581738784	41.41198953	67.2973
Glass	0.5989779839	0.1133040132	0.5452416711	0.07414
Ionosphere	0.1090565498	0.05361442094	0.1213389024	0.08060
Iris	0.006729922081	0.01249749253	0.05668451999	0.05109
LandCover	0.720098402	0.3375288596	3.42501748	2.4024
MNIST	1.186380063	0.611684428	6.840124999	4.6642
Music	785.1514395	42.1096223	1902.920986	225.9133
Musk	0.06887121318	0.01606985477	0.3835405492	0.1705
Parkinsons	0.06120370054	0.06859344536	0.07586307128	0.0648
Residential	510846.4153	390287.4549	1672532.72	90914.9714
Scadi	0.2522075711	0.2881453386	0.4377157515	0.2744
Seisures	0.1453737175	0.106864704	0.4626022419	0.1188
Servo	0.7976142915	0.8103341331	33.11948728	57.6692
Sonar	0.06507311849	0.06919359194	0.1865161968	0.1615676093
Soybean	0.2345737889	0.08279427322	0.8928987318	0.4577
Spambase	0.1081700663	0.00867025515	0.2325407487	0.0692
Thyroid	0.01946807243	0.007756024899	0.119318722	0.09439
Wine	0.006205653102	0.01352924571	0.0002134919572	0.0012
Zoo	0.01051604764	0.01621362823	0.08378003993	0.1467

Table A.28: Training Loss Results

	Average	StdDev	Average	StdDev
TrainingLoss	ACO	ACO	Bat	Bat
Auto	23.65286145	21.67998583	22.47689557	3.125577209
BreastCancer	0.0019	0.0029	0	0
ComputerHardware	28896.99815	5916.488335	2822.231304	1184.194751
Facebook	5.140814175	3.009882316	6.008339272	2.419785515
ForestFires	5.140814175	3.009882316	6.008339272	2.419785515
Glass	0.7826168948	0.1957387776	0.7868469567	0.4361486272
Ionosphere	0.1836437988	0.09219246475	0.1435072209	0.08711417867
Iris	0.2834058306	0.2642492461	0.08159263381	0.06860795023
LandCover	0.3644232383	0.2573623646	1.584489971	0.4720598541
MNIST	1.363764105	0.5301356211	2.435537364	0.2117669911
Music	1610.105228	292.496602	1102.167173	108.3983266
Musk	0.08459103938	0.03224391163	0.3039040001	0.2249004136
Parkinsons	0.08408882756	0.07504711047	0.1582559534	0.1027686471
Residential	1575294.303	108906.3713	1048298.445	536209.5927
Scadi	0.02213554332	0.03792131735	0.5734814362	0.4324532757
Seisures	0.3808630826	0.08848567019	0.3329244514	0.1058358423
Servo	1.315800001	0.7092599281	0.8423919051	0.2312331991
Sonar	0.06975088272	0.06905918477	0.2140056729	0.1677642286
Soybean	1.079416807	0.5788309057	0.60080836	0.2058815799
Spambase	0.142365953	0.03244517759	0.2183711781	0.04025331681
Thyroid	0.1659948076	0.04517586052	0.1414639456	0.08764896609
Wine	0.08737963448	0.1408635288	0.02478542568	0.03692482394
Zoo	0.1668307459	0.2558764804	0.05854872118	0.0715572995

Table A.29: Training Loss Results

	Average	StdDev	Average	StdDev
TrainingLoss	BFA	BFA	Firefly	Firefly
Auto	35.90513794	23.0923726	501.9466604	391.1458185
BreastCancer	0	0	0	0
ComputerHardware	10148.08808	6055.111834	36982.06274	4829.626779
Facebook	225.4043148	353.4266457	218.7106874	359.3800488
ForestFires	225.4043148	353.4266457	218.7106874	359.3800488
Glass	1.350335789	0.3295170335	0.5395519835	0.2696322578
Ionosphere	0.4238675605	0.2045696591	0.09549292523	0.08875246122
Iris	0.2088251773	0.2758286371	0.02071551561	0.01901642219
LandCover	6.155434305	3.237536873	0.7199630495	0.1583906498
MNIST	8.101744331	5.029606181	1.016749985	0.5368352111
Music	1405.304559	386.4573771	2310.55763	244.7975533
Musk	0.4955071721	0.1950628132	0.07672633264	0.009887421896
Parkinsons	0.4232210154	0.3039603478	0.007353392975	0.008296680865
Residential	1076340.116	279048.4833	1679054.226	82572.35123
Scadi	1.481509212	0.4901028508	0.2965422842	0.4003961394
Seisures	0.7089136914	0.5338058925	0.2262082571	0.1541336779
Servo	95.25167267	164.7496623	165.6121968	224.7901686
Sonar	0.9155110878	0.5614922569	0.1070387428	0.1021317109
Soybean	1.821403579	0.783534324	0.326391863	0.1632772652
Spambase	1.006173704	0.5095219525	0.1414856161	0.01084555944
Thyroid	0.1981550012	0.08432889826	0.06466482178	0.01594199548
Wine	0.259965134	0.3042272157	0.0003188492836	0.0006417757634
Zoo	0.5684441639	0.4617809558	0.00003571029656	0.00008031901346

Table A.30: Training Loss Results

	Average	StdDev
TrainingLoss	Fish	Fish
Auto	44.31697837	8.859135933
BreastCancer	0	0
ComputerHardware	28095.09071	8108.104538
Facebook	6.152534343	0.8758083357
ForestFires	6.152534343	0.8758083357
Glass	0.6886202728	0.06140126159
Ionosphere	0.1484647435	0.04726320641
Iris	0.05594129766	0.03308574395
LandCover	2.824104532	1.505803876
MNIST	4.793939207	1.59972957
Music	1765.511709	147.592887
Musk	0.2719675295	0.07707986557
Parkinsons	0.1045495446	0.02459392543
Residential	1609191.523	121061.2885
Scadi	0.1058057862	0.07533300204
Seisures	0.4114710613	0.07977494689
Servo	1.670977908	0.4900745422
Sonar	0.4145303194	0.2423518552
Soybean	1.266179187	0.1460633865
Spambase	0.2542504144	0.08591959839
Thyroid	0.1003555085	0.01418226242
Wine	0.001682766458	0.001386877348
Zoo	0.04607056067	0.03436679082

Table A.31: Training Loss Results

	Average	StdDev	Average	StdDev
TestingLoss	PSO	PSO	ABC	ABC
BreastCancer	1	0	1	0
Glass	0.7677387913	0.04464493213	0.7876705653	0.03108283704
Ionosphere	0.5448611819	0.008578547635	0.5469483418	0.01009619823
Iris	0.9973611111	0.00872977415	0.9788888889	0.01794740651
LandCover	0.1378503906	0.002531561882	0.1382716166	0.009974488469
MNIST	0.7986140278	0.06684611556	0.2264323611	0.08212618794
Musk	0.977797777	0.008340788673	0.8822565366	0.02335538282
Parkinsons	0.9793803419	0.02785191955	0.9757478633	0.03704378861
Scadi	0.9261904762	0.09032397033	0.884672619	0.07710766843
Seisures	0.9469528986	0.04339609287	0.8412355072	0.02373137249
Sonar	0.986997992	0.01763646016	0.9404116466	0.0651315228
Soybean	0.9350340136	0.02925966286	0.7514285714	0.1435975578
Spambase	0.9628238225	0.005390472443	0.9206000906	0.02221968773
Thyroid	0.8587404617	0.004270380322	0.8828692063	0.02686736864
Wine	0.9994131455	0.003486435081	1	0
Zoo	0.9995833333	0.002253226756	0.97875	0.03981311384

Table A.32: Training Accuracy Results

	Average	StdDev	Average	StdDev
TestingLoss	ACO	ACO	Bat	Bat
BreastCancer	1	0	1	0
Glass	0.6997563353	0.07560438441	0.7081871345	0.1584068007
Ionosphere	0.5481135204	0.009468530909	0.5469804422	0.008604085713
Iris	0.9123611112	0.1191022079	0.9673611111	0.03072295205
LandCover	0.1390692354	0.003159588623	0.1340377288	0.01676248982
MNIST	0.7227902777	0.1565837756	0.46214375	0.1993836218
Musk	0.9747710623	0.01176835848	0.8304360869	0.1843751716
Parkinsons	0.9735042736	0.04474435146	0.9345619658	0.05298753046
Scadi	1	0.003260253345	0.8519345238	0.1095272891
Seisures	0.8588432971	0.03486493992	0.8623586956	0.04038609451
Sonar	0.9902108433	0.0221660682	0.9197289157	0.08956199454
Soybean	0.7147959183	0.1735481543	0.8092176871	0.0781864468
Spambase	0.9500747283	0.01160981784	0.9206453804	0.01617060001
Thyroid	0.8999281341	0.01525257141	0.8812463095	0.01921530784
Wine	0.9914906103	0.01557119435	0.99342723	0.01327084696
Zoo	0.9754166667	0.04524276545	0.9904166667	0.01933070182

Table A.33: Training Accuracy Results

	Average	StdDev	Average	StdDev
TestingLoss	BFA	BFA	Firefly	Firefly
BreastCancer	1	0	1	0
Glass	0.4982456141	0.1118830653	0.7973196881	0.1015549163
Ionosphere	0.534098356	0.07546725138	0.5446832483	0.009752183903
Iris	0.8707407408	0.1521908485	0.9929861111	0.007108587757
LandCover	0.1369864638	0.02050415696	0.1399196042	0.003355003465
MNIST	0.2301662963	0.1247223567	0.6853247223	0.1980636585
Musk	0.8298261126	0.07818412681	0.9738505747	0.008240103237
Parkinsons	0.7998575499	0.0919561048	0.9998397436	0.007658716631
Scadi	0.5928571429	0.1594758129	0.9357142857	0.09324648267
Seisures	0.7947669082	0.04145731893	0.9114818841	0.06265478626
Sonar	0.6567603748	0.1253647099	0.9713855422	0.03164592549
Soybean	0.4977324263	0.2507616032	0.9168027211	0.0580107387
Spambase	0.7321346619	0.06088055482	0.9495606884	0.004526397779
Thyroid	0.9009389798	0.02253483482	0.8648402256	0.003027408657
Wine	0.8730829421	0.128339245	1	0
Zoo	0.8008333333	0.1815228101	1	0

Table A.34: Training Accuracy Results

	Average	StdDev
TestingLoss	Fish	Fish
BreastCancer	1	0
Glass	0.7331871345	0.03075760361
Ionosphere	0.5451855867	0.008230605043
Iris	0.9828472222	0.009901475497
LandCover	0.1418702091	0.01150548424
MNIST	0.3654558333	0.1415818791
Musk	0.907488632	0.01780772898
Parkinsons	0.9664529915	0.0197192652
Scadi	0.9791666666	0.02553549828
Seisures	0.8355326087	0.02523702131
Sonar	0.83062249	0.1115378943
Soybean	0.6178571429	0.05053821066
Spambase	0.9106453804	0.02693739924
Thyroid	0.8753848443	0.005376452603
Wine	1	0
Zoo	0.9977083333	0.005389094291

Table A.35: Training Accuracy Results

	Average	Stddev	Average	StdDev
	PSO	PSO	ABC	ABC
Auto	16.44742	4.36273	33.05582	16.80204
BreastCancer	0.00000	0.00000	0.00000	0.00006
ComputerHardware	9389.60304	8831.84026	24533.82525	18761.80714
Facebook	82934.96446	118948.05812	169519.45225	163699.98266
ForestFires	5.43279	1.30394	43.83243	67.10829
Glass	1.05688	0.27580	0.96918	0.22374
Ionosphere	0.40632	0.16902	0.35401	0.13835
Iris	1.15930	1.15060	0.25373	0.41321
LandCover	2.86336	2.02638	4.69036	3.50257
MNIST	1.29834	0.68419	6.84358	4.67091
Music	1294.70239	129.45785	1987.35971	231.26325
Musk	0.14809	0.05390	0.45255	0.22699
Parkinsons	0.51798	0.43516	0.25666	0.12025
Residential	601187.71299	369043.24371	1706264.67623	327655.01382
Scadi	2.68415	2.01800	1.61487	1.05410
Seisures	0.19714	0.10442	0.53930	0.17728
Servo	1.18205	0.96263	33.53540	58.21258
Sonar	1.03818	0.68483	0.82260	0.46316
Soybean	0.86867	0.27665	1.23176	0.40283
Spambase	0.19241	0.02327	0.28386	0.07266
Thyroid	0.07482	0.01375	0.14026	0.08581
Wine	0.45171	0.62776	0.32260	0.51809
Zoo	0.97831	0.95761	0.45076	0.41718

Table A.36: Testing Loss Results

	Average	StdDev	Average	StdDev
	ACO	ACO	Bat	Bat
Auto	26.47308	20.55221517	23.84570795	5.737863788
BreastCancer	0.00187	0.002864121833	0.00000003625	0.000000397
ComputerHardware	30865.67421	18648.62755	7741.384151	7087.744309
Facebook	157781.89999	152686.049	82526.91278	120859.0274
ForestFires	6.86291	4.557795484	6.813285775	3.003587378
Glass	1.02041	0.2016268038	1.133578509	0.3885099502
Ionosphere	0.32451	0.09986648374	0.3541868334	0.1166258088
Iris	0.30156	0.2661660461	0.3106367646	0.6229557531
LandCover	2.65337	1.97912908	2.052601072	0.4051935939
MNIST	1.48547	0.6053224724	2.459850382	0.2381860902
Music	1768.88205	246.4134045	1415.159263	150.0972819
Musk	0.18664	0.1214191468	0.3274157607	0.2150442408
Parkinsons	0.26976	0.1363728225	0.3830811671	0.1859466339
Residential	1647234.36806	362587.8884	1078490.462	530933.3439
Scadi	1.54477	1.269573166	2.126106945	1.567713792
Seisures	0.48810	0.07936146489	0.3521479053	0.1032495622
Servo	1.61566	0.6887017586	1.277920635	0.5238680936
Sonar	0.73532	0.4532562358	0.7768549158	0.3210949263
Soybean	1.36022	0.4856930321	1.223999783	0.3640427569
Spambase	0.22417	0.05432704135	0.2555007224	0.04689564405
Thyroid	0.18740	0.03525791543	0.1664728102	0.07788513623
Wine	0.14225	0.1369259671	0.2739303273	0.5262642899
Zoo	0.41243	0.287375947	0.9241296241	1.123416363

Table A.37: Testing Loss Results

	Average	StdDev	Average	StdDev
	BFA	BFA	Firefly	Firefly
Auto	37.31266138	23.66558076	504.7029917	394.550809
BreastCancer	0	0	0	0
ComputerHardware	12613.24285	10711.54564	32800.14362	17382.15353
Facebook	74639.01774	111221.914	169326.6842	151567.4045
ForestFires	221.8481419	344.0157667	217.6514909	355.9657174
Glass	1.500709807	0.4423603046	0.9766345506	0.2152326805
Ionosphere	0.4987464465	0.1782651545	0.3981156886	0.1764671762
Iris	0.5669264308	0.6977562703	0.4256194707	0.6648034964
LandCover	6.783567295	3.740148744	1.317425919	0.6725732002
MNIST	8.116289427	5.053495911	1.036273904	0.5326150473
Music	1577.159996	273.488676	2345.616339	284.0858181
Musk	0.5372264485	0.1895170715	0.1191512747	0.01889781506
Parkinsons	0.5611707577	0.3863247392	0.4862181421	0.5458522201
Residential	1082993.473	283449.8573	1673597.358	301097.5084
Scadi	3.227785055	2.246319712	1.474655583	0.9215401342
Seisures	0.7264799743	0.5421752079	0.2718531864	0.151815628
Servo	96.17848192	165.6080929	167.3426093	226.4019625
Sonar	1.392194101	0.9568875911	0.6979331135	0.3861722979
Soybean	2.193642603	0.5750982174	0.7202572188	0.2217441987
Spambase	1.033831312	0.5257460343	0.196349729	0.02120883768
Thyroid	0.2121358167	0.07119464377	0.1031868357	0.0210379912
Wine	0.4895841475	0.3766228715	0.2001874331	0.2625440659
Zoo	1.720406327	1.180646223	0.9802421492	1.066458844

Table A.38: Testing Loss Results

	Average	StdDev
	Fish	Fish
Auto	43.32664656	10.80599596
BreastCancer	0	0
ComputerHardware	29050.93382	19302.66545
Facebook	167951.8093	158234.5777
ForestFires	6.649343888	1.365778413
Glass	0.9231690134	0.1657014197
Ionosphere	0.3265597358	0.1040562765
Iris	0.1051039244	0.08210090956
LandCover	3.78341766	2.334959591
MNIST	4.815497934	1.611746234
Music	1785.199204	179.4487113
Musk	0.3005975805	0.09325447043
Parkinsons	0.3002885975	0.1435812261
Residential	1652620.37	359536.5452
Scadi	1.596538253	0.9971905151
Seisures	0.4259418231	0.08169869112
Servo	1.907408687	0.707790057
Sonar	0.8815660692	0.3402215814
Soybean	1.582757145	0.1932229469
Spambase	0.2830057713	0.09056460165
Thyroid	0.1248201294	0.0149004696
Wine	0.1069263371	0.09562161277
Zoo	0.3007676898	0.2862468241

Table A.39: Testing Loss Results

	PSO	PSO	ABC	ABC
BreastCancer	1	0	1	0
Glass	0.6498062018	0.06777802196	0.6616279072	0.06624526141
Ionosphere	0.5579960987	0.02980603475	0.5609237585	0.03406792975
Iris	0.9444444445	0.04018998615	0.9502777778	0.03614539069
LandCover	0.1388271605	0.007444302307	0.1361883859	0.01213921371
MNIST	0.7891083333	0.06401055572	0.2261716667	0.08145012132
Musk	0.9503030303	0.01275099245	0.8683775253	0.02312119273
Parkinsons	0.8933760683	0.05359942166	0.8961538461	0.04831949384
Scadi	0.7160714286	0.1169100817	0.650595238	0.1572574067
Seisures	0.9249492754	0.03887474546	0.828865942	0.020042234
Sonar	0.7728174603	0.06068266358	0.7488095238	0.06241924102
Soybean	0.7673387097	0.06829097935	0.6470430108	0.1219311954
Spambase	0.932899023	0.007981194102	0.9064332248	0.02018237516
Thyroid	0.8587798025	0.005494941256	0.8848667264	0.02822662671
Wine	0.9703703702	0.02909924707	0.9687499999	0.02834009122
Zoo	0.9075396826	0.06029578096	0.9162698412	0.06997884085

Table A.40: Testing Accuracy Results

	Average	StdDev	Average	StdDev
	ACO	ACO	Bat	Bat
BreastCancer	1	0	1	0
Glass	0.6029069769	0.09694657184	0.6062015505	0.1295108228
Ionosphere	0.5556552933	0.03201729206	0.5545493619	0.03076941378
Iris	0.8919444445	0.1353994907	0.9469444445	0.04417430745
LandCover	0.140269776	0.008804406289	0.1335793324	0.01777188719
MNIST	0.7126541667	0.152432935	0.4603	0.1971666135
Musk	0.9478977273	0.01897024054	0.8228093435	0.1812061351
Parkinsons	0.8931623931	0.05066378357	0.8564102564	0.06107555298
Scadi	0.7922619048	0.1029421974	0.6017857143	0.1848525193
Seisures	0.8362463768	0.02777840018	0.8550688406	0.03701874862
Sonar	0.7900793651	0.06901575993	0.7444444445	0.07184723475
Soybean	0.6252688172	0.1475694548	0.6681451614	0.08757093351
Spambase	0.925190011	0.01055485844	0.907908071	0.01780276783
Thyroid	0.9019104076	0.01564337077	0.8827218368	0.01988094509
Wine	0.9696759258	0.03064137849	0.9472222221	0.04196451784
Zoo	0.9059523809	0.06776543593	0.8972222222	0.07083587618

Table A.41: Testing Accuracy Results

	Average	StdDev	Average	StdDev
	BFA	BFA	Firefly	Firefly
BreastCancer	1	0	1	0
Glass	0.45374677	0.1204125521	0.6531007754	0.07984058144
Ionosphere	0.5444003615	0.06590524009	0.5595731668	0.03380518406
Iris	0.8503703704	0.1597594947	0.9561111112	0.03855465282
LandCover	0.1341795458	0.0264040404	0.1411188843	0.007167814789
MNIST	0.2298277778	0.1240395082	0.6810816667	0.1971291307
Musk	0.8209090909	0.07614367478	0.9561805555	0.009447112556
Parkinsons	0.7672364672	0.09371935432	0.9	0.0541536045
Scadi	0.4301587302	0.1769347144	0.7273809524	0.1190676254
Seisures	0.792531401	0.04085710086	0.8941775363	0.05644275593
Sonar	0.6150793651	0.1019709321	0.7980158731	0.05706576045
Soybean	0.4304659499	0.2120029463	0.779032258	0.076787655
Spambase	0.7290901876	0.05941206359	0.930311256	0.007296513369
Thyroid	0.902728839	0.02302623894	0.8649143347	0.005054309091
Wine	0.837037037	0.1259070296	0.9743055554	0.02282177333
Zoo	0.7158730159	0.1728555305	0.9055555555	0.06249932501

Table A.42: Testing Accuracy Results

	Average	StdDev
	Fish	Fish
BreastCancer	1	0
Glass	0.6587209305	0.06579669501
Ionosphere	0.5661178337	0.03362056026
Iris	0.9619444445	0.03647320706
LandCover	0.1400050297	0.01196108703
MNIST	0.3645075	0.1416982446
Musk	0.9010732323	0.01789977945
Parkinsons	0.8835470085	0.04969722505
Scadi	0.7107142857	0.1507306925
Seisures	0.8322862319	0.02484684323
Sonar	0.7178571428	0.07740947879
Soybean	0.5333333334	0.06826352679
Spambase	0.9014205574	0.0264988788
Thyroid	0.8775431912	0.006239857018
Wine	0.966898148	0.02807685299
Zoo	0.9194444444	0.06410768783

Table A.43: Testing Accuracy Results

	Average	StdDev	Average	StdDev
Saturation	PSO	PSO	ABC	ABC
Auto	0.9962090281	0.00411489772	0.9975780041	0.001158125449
BreastCancer	0.7643404477	0.1467644817	0.6831285635	0.1499837987
ComputerHardware	0.8163866014	0.2003517914	0.613399562	0.1077015117
Facebook	0.9031677355	0.1081577164	0.6653875951	0.09654495848
ForestFires	0.6691014541	0.08896031344	0.6229850303	0.0976227133
Glass	0.7255243913	0.07273439254	0.6882333705	0.08318268772
Ionosphere	0.8521480741	0.09594164733	0.8021846119	0.08569775796
Iris	0.8807837132	0.0566479185	0.7366534925	0.1094503133
LandCover	0.886401257	0.07553326671	0.8813585345	0.07915948212
MNIST	0.9520130995	0.04209493053	0.9302303505	0.05991142404
Music	0.9493804651	0.02873139826	0.8394174087	0.08954988193
Musk	0.9155999777	0.05421888911	0.8917369316	0.07209570998
Parkinsons	0.8034147925	0.06274858562	0.7487416603	0.08884842338
Residential	0.9832135407	0.01176102838	0.8782100631	0.08946056715
Scadi	0.9115827596	0.02905522278	0.8646149235	0.07247240323
Seisures	0.8827633123	0.063339149	0.8014062845	0.1061359194
Servo	0.8226174759	0.09396491682	0.8173929016	0.1117677516
Sonar	0.8655416232	0.09216660927	0.8398111109	0.08727104252
Soybean	0.9129962636	0.05163458616	0.8565393198	0.07197349631
Spambase	0.8383506042	0.07096213555	0.7673402705	0.08421222267
Thyroid	0.8592805463	0.06745260534	0.7119735699	0.1345103193
Wine	0.8816246114	0.06621959633	0.8206145439	0.07286324485
Zoo	0.896222658	0.05222464417	0.8194998809	0.08030063445

Table A.44: Saturation Results

	Average	StdDev	Average	StdDev
Saturation	ACO	ACO	Bat	Bat
Auto	0.9911830455	0.01150617013	0.999062293	0.001441605607
BreastCancer	0.6139275429	0.1513623137	0.8125544638	0.1194208885
ComputerHardware	0.5223343553	0.09318142368	0.9508183317	0.02082450724
Facebook	0.5967966571	0.08876397806	0.9598758552	0.02820762163
ForestFires	0.588607729	0.1320466029	0.6458854645	0.07836776436
Glass	0.5996858439	0.1076870469	0.7303119699	0.1026120513
Ionosphere	0.7696260362	0.08824279795	0.8355636448	0.07537880493
Iris	0.7364960465	0.08665786738	0.8761371557	0.08517506897
LandCover	0.8981478155	0.07872894503	0.883990502	0.1191301703
MNIST	0.9518719544	0.04994368903	0.9043312201	0.07299518444
Music	0.8113411068	0.09118231903	0.9554294445	0.0211796865
Musk	0.911100727	0.05946082441	0.9216839479	0.07183466565
Parkinsons	0.7289404739	0.09325376094	0.7654864968	0.1113334366
Residential	0.9421327018	0.04926155699	0.9920500824	0.005635404537
Scadi	0.9138566013	0.04759521053	0.8938314981	0.03277688122
Seisures	0.8653135774	0.06182150956	0.8519895693	0.08702691939
Servo	0.8044027835	0.115697508	0.8424682237	0.05752837028
Sonar	0.8506798885	0.06482163954	0.8244016183	0.1125921817
Soybean	0.8401620878	0.07143756152	0.9228657413	0.03338771059
Spambase	0.7584137478	0.09311141087	0.8213393597	0.09476539224
Thyroid	0.581174637	0.09980657839	0.7593344201	0.06949036616
Wine	0.7684656923	0.09083365559	0.8515966506	0.08008266546
Zoo	0.8051971524	0.08265167502	0.8805748577	0.09620194935

Table A.45: Saturation Results

	Average	StdDev	Average	StdDev
Saturation	BFA	BFA	Firefly	Firefly
Auto	0.999370579	0.000767953857	0.9990140459	0.001788447243
BreastCancer	0.8733683537	0.1189700543	0.789151377	0.1210640231
ComputerHardware	0.9054810533	0.0446670634	0.5156234596	0.1020660927
Facebook	0.9403346425	0.03980777122	0.6703198741	0.09333825952
ForestFires	0.6669534897	0.0808395322	0.5265677066	0.1674438999
Glass	0.6749980562	0.07594787552	0.6524891676	0.09388627933
Ionosphere	0.8547750951	0.1074807357	0.812981741	0.1398833292
Iris	0.9493062052	0.03689997544	0.8858379019	0.07629973
LandCover	0.9034084374	0.09434395284	0.7334651101	0.1392687364
MNIST	0.9400207899	0.06768801171	0.8829466000000	0.0870847424280
Music	0.9499367751	0.02340637107	0.8963941976	0.03026827837
Musk	0.9119028558	0.06152900526	0.844393643	0.0676443942
Parkinsons	0.7608862988	0.09622116358	0.8315009567	0.09172029859
Residential	6.874578563	10.3687971	0.9423383767	0.06148758939
Scadi	0.9225905114	0.05749168197	0.8268775941	0.09940830088
Seisures	0.8543845414	0.1327050326	0.7107511438	0.1717320621
Servo	0.8380714408	0.1096604206	0.8302542356	0.1198184062
Sonar	0.863433754	0.108316244	0.7494777435	0.160376792
Soybean	0.8698614908	0.06801222999	0.8742742913	0.06576689104
Spambase	0.8787724259	0.08052718197	0.7119668844	0.07868310632
Thyroid	0.7952945538	0.1335859603	0.7315484248	0.1157661877
Wine	0.8916329914	0.0722095328	0.8644061875	0.05303254934
Zoo	0.930386749	0.05808055721	0.9114934673	0.05811916828

Table A.46: Saturation Results

	Average	StdDev
Saturation	Fish	Fish
Auto	0.8807044602	0.05590847245
BreastCancer	0.7652604784	0.09627319169
ComputerHardware	0.5772769272	0.0756429874
Facebook	0.7132787591	0.06462346989
ForestFires	0.6348665638	0.05068106419
Glass	0.6340736874	0.08220280803
Ionosphere	0.8157125259	0.04970438936
Iris	0.7800302802	0.07142169869
LandCover	0.9154475368	0.02887060258
MNIST	0.9188801463	0.04667717583
Music	0.8797428544	0.03384858079
Musk	0.9121922917	0.02626453948
Parkinsons	0.7721272302	0.05664050097
Residential	0.9265116414	0.03269570108
Scadi	0.8868723861	0.0618965181
Seisures	0.8558539171	0.0507541411
Servo	0.8619694437	0.04680852563
Sonar	0.8680464774	0.03368717697
Soybean	0.8308038113	0.06247436417
Spambase	0.7856479769	0.05061195365
Thyroid	0.6481177324	0.07972710949
Wine	0.8020637657	0.06772494533
Zoo	0.8046676836	0.07289653642

Table A.47: Saturation Results

	Average	StdDev	Average	StdDev
Overfitting	PSO	PSO	ABC	ABC
Auto	12.91684092	4.647225019	31.74457177	18.0409326
BreastCancer	0	0	0	0
ComputerHardware	5670.652546	7942.745773	20733.20535	9969.542486
Facebook	3.530744963	0.6251671628	41.38840061	67.31665246
ForestFires	3.530744963	0.6251671628	41.38840061	67.31665246
Glass	0.597930541	0.1135414265	0.5467836618	0.07491778247
Ionosphere	0.1085724071	0.05380094094	0.1217161669	0.08036607301
Iris	0.008251085265	0.02032764463	0.05681463605	0.05346565399
LandCover	0.731558474	0.3549191365	3.409416024	2.418320685
MNIST	1.218353675	0.685773476	6.807886134	4.693786635
Music	796.5526268	134.2401107	1900.385495	225.2203121
Musk	0.07019185367	0.02155826919	0.3821704201	0.172650011
Parkinsons	0.06209022977	0.06848316814	0.07539168727	0.06481773152
Residential	523213.3232	406504.6505	1672122.807	89758.4814
Scadi	0.2567222564	0.2884045938	0.4332010858	0.2770923886
Seisures	0.1475397578	0.1084832373	0.4630479995	0.1187439504
Servo	0.8020171008	0.811255211	33.11948099	57.6693455
Sonar	0.0655508894	0.06947088666	0.1854700099	0.1623112259
Soybean	0.2385045659	0.09070465323	0.8935585227	0.4574785834
Spambase	0.1090017748	0.01236979392	0.2319148829	0.06979581505
Thyroid	0.02096307943	0.01740434883	0.1190320479	0.09319108177
Wine	0.00620565319	0.01352924567	0.0003129920772	0.001816301263
Zoo	0.01052265162	0.01620946981	0.08412615333	0.1465805495

Table A.48: Overfitting Indicator Values

	Average	StdDev	Average	StdDev
Overfitting	ACO	ACO	Bat	Bat
Auto	23.73089835	21.64789109	22.50693428	3.114401316
BreastCancer	0	0	0	0
ComputerHardware	28661.08009	6384.058365	2875.244196	1307.279555
Facebook	5.148782733	3.005960006	6.102945931	2.564318272
ForestFires	5.148782733	3.005960006	6.102945931	2.564318272
Glass	0.7812806172	0.1968848991	0.7918002392	0.4366370518
Ionosphere	0.1835593259	0.09227018482	0.1460331676	0.09002370867
Iris	0.2818445515	0.2653085326	0.08169541443	0.06851017438
LandCover	0.3809852743	0.2984269322	1.627063206	0.6930242804
MNIST	1.379483032	0.5394523605	2.470929806	0.4571392664
Music	1602.424721	297.4811118	1103.357046	107.5562143
Musk	0.08550130347	0.03312744436	0.306791888	0.2254054752
Parkinsons	0.08580010919	0.07658464235	0.1590242191	0.1038382799
Residential	1573251.839	108316.3827	1048352.861	536254.391
Scadi	0.02338172478	0.03960751174	0.5789747571	0.4312078819
Seisures	0.3790286401	0.09014908872	0.3338681627	0.1050487535
Servo	1.311459002	0.7113716072	0.841053685	0.2324642265
Sonar	0.07376006362	0.07874925773	0.2119320876	0.1658581506
Soybean	1.077644589	0.5802499524	0.6144182514	0.249354186
Spambase	0.1427377277	0.03255526504	0.2202481573	0.0433203144
Thyroid	0.1654220054	0.04573560407	0.1419017926	0.08750719332
Wine	0.08728823758	0.140916436	0.02497987899	0.03685975048
Zoo	0.1675774419	0.2556407305	0.05917689483	0.07253113793

Table A.49: Overfitting Indicator Results

	Average	StdDev	Average	StdDev
Overfitting	BFA	BFA	Firefly	Firefly
Auto	43.09670812	80.95924326	494.9673032	391.7047748
BreastCancer	0	0	0	0
ComputerHardware	10425.55145	6723.318594	36801.00996	5035.475454
Facebook	225.5600819	353.3374676	218.4699848	359.5143155
ForestFires	225.5600819	353.3374676	218.4699848	359.5143155
Glass	1.343822935	0.3414327086	0.5415994092	0.2690951085
Ionosphere	0.4232327443	0.2047702821	0.09418606251	0.08652596304
Iris	0.2086324461	0.2759675217	0.02108516646	0.01904158725
LandCover	6.102660559	3.272475042	0.7226685646	0.1629191091
MNIST	8.049664172	5.074558758	1.056978143	0.6713659545
Music	1416.829339	397.6853871	2307.36505	245.0495371
Musk	0.4918864388	0.1986474926	0.07813119432	0.01878532699
Parkinsons	0.4204567062	0.30615645	0.008498051442	0.01565916544
Residential	1077981.529	282061.4385	1679515.097	82767.44808
Scadi	1.474782126	0.5047363943	0.296927227	0.4001322114
Seisures	0.7071168188	0.5355444717	0.2295616931	0.1554554751
Servo	95.34223547	164.7001468	165.5364419	224.8440946
Sonar	0.9137104046	0.5640156326	0.1080580433	0.1019114673
Soybean	1.805500325	0.7961722016	0.3342950287	0.1809056452
Spambase	1.003847824	0.5129395155	0.1421266426	0.01182577629
Thyroid	0.1973700893	0.08530896023	0.06490125279	0.01586773432
Wine	0.2597625774	0.3043934835	0.0003334779669	0.0006543308435
Zoo	0.5667168377	0.4635275237	0.0001163637216	0.0008867529252

Table A.50: Overfitting Indicator Results

	Average	StdDev
Overfitting	Fish	Fish
Auto	44.28091372	8.887744865
BreastCancer	0	0
ComputerHardware	28159.19215	8111.795822
Facebook	6.159570638	0.8760987956
ForestFires	6.159570638	0.8760987956
Glass	0.6894209336	0.0610285687
Ionosphere	0.1483300427	0.04743991472
Iris	0.05603858545	0.03320840063
LandCover	2.838170049	1.504233728
MNIST	4.788392918	1.605334743
Music	1762.715704	144.990345
Musk	0.2721372953	0.07738325376
Parkinsons	0.1041423505	0.02428830268
Residential	1608078.5	120955.03
Scadi	0.1062941925	0.07546050262
Seisures	0.4108145423	0.0797860534
Servo	1.665466214	0.4883974569
Sonar	0.4167797676	0.2421153097
Soybean	1.266859746	0.1464898067
Spambase	0.2547276695	0.08612301599
Thyroid	0.1005768579	0.01403252731
Wine	0.001682155739	0.001392725345
Zoo	0.04637611403	0.0343480268

Table A.51: Overfitting Indicator Results

Algorithm	PSO	ABC	ACO	BA	BFA	Firefly	Fish
Auto	1	4	2	3	5	7	6
ComputerHardware	1	4	6	2	3	7	5
Facebook	1	5	2	3	6	7	4
ForestFires	1	5	2	3	6	7	4
Music	1	6	4	2	3	7	5
Residential	1	6	4	2	2	6	5
Servo	1	5	3	2	4	7	6
Average Rank	1	5	3.2857	2.4286	4.1429	6.8571	5
Rank	1	5	3	2	4	7	5
First Frequency	7	0	0	0	0	0	0

Table A.52: Training Loss Regression Datasets

Algorithm	PSO	ABC	ACO	BA	BFA	Firefly	Fish
BreastCancer	1	6	7	1	1	1	5
Glass	3	2	6	4	7	1	5
Ionosphere	2	3	6	4	7	1	5
Iris	1	3	7	5	6	2	4
LandCover	2	6	1	4	7	3	5
MNIST	2	6	3	4	7	1	5
Musk	1	6	2	4	7	3	5
Parkinsons	2	3	4	6	7	1	5
Scadi	2	6	1	5	7	4	3
Seizures	1	6	4	3	7	2	5
Sonar	1	4	2	5	7	3	6
Soybean	1	4	5	3	7	2	6
Spambase	1	4	2	4	6	3	5
Thyroid	1	3	6	5	7	2	3
Wine	1	1	6	5	7	3	4
Zoo	2	3	5	4	7	1	6
Average Rank	1.5	4.125	4.1875	4.125	6.5	2.0625	4.8125
Rank	1	3	5	3	7	2	6
First Frequency	9	1	2	1	1	6	0

Table A.53: Training Loss Classification Datasets

Algorithm	PSO	ABC	ACO	BA	BFA	Firefly	Fish
Breast Cancer	1	1	1	1	1	1	1
Glass	3	2	6	4	7	1	5
Ionosphere	5	6	2	1	3	7	4
Iris	1	4	7	5	6	2	3
LandCover	3	6	2	5	7	1	4
MNIST	1	6	2	4	7	3	5
Musk	1	6	2	4	7	3	5
Parkinsons	4	3	2	6	7	1	5
Scadi	4	5	1	6	7	2	2
Seizures	1	5	4	3	7	2	6
Sonar	2	4	1	5	7	3	6
Soybean	1	4	5	3	7	2	6
Spambase	1	4	2	5	7	3	6
Thyroid	7	5	1	3	2	6	4
Wine	3	1	6	2	7	3	3
Zoo	6	2	2	4	7	1	5
Average Rank	2.75	4	2.875	3.8125	6	2.5625	4.375
Rank	2	5	3	4	7	1	6
First Frequency	7	2	4	2	1	5	1

Table A.54: Training Accuracy Classification Datasets

Algorithm	PSO	ABC	ACO	BA	BFA	Firefly	Fish
Auto	1	4	2	3	5	7	6
BreastCancer	1	6	7	1	1	1	5
ComputerHardware	1	4	6	2	3	7	5
Facebook	1	5	2	3	6	7	4
ForestFires	1	5	2	3	6	7	4
Glass	3	2	6	4	7	1	5
Ionosphere	2	3	6	4	7	1	5
Iris	1	3	7	5	6	2	4
LandCover	2	6	1	4	7	3	5
MNIST	2	6	3	4	7	1	5
Music	1	6	4	2	3	7	5
Musk	1	6	2	4	7	3	5
Parkinsons	2	3	4	6	7	1	5
Residential	1	6	4	2	2	6	5
Scadi	2	6	1	5	7	4	3
Seizures	1	6	4	3	7	2	5
Servo	1	5	3	2	4	7	6
Sonar	1	4	2	5	7	3	6
Soybean	1	4	5	3	7	2	6
Spambase	1	4	2	4	6	3	5
Thyroid	1	3	6	5	7	2	3
Wine	1	1	6	5	7	3	4
Zoo	2	3	5	4	7	1	6
Average	1.3478	4.3913	3.9130	3.6087	5.7826	3.5217	4.8696
Rank	1	5	4	3	7	2	6
First Frequency	16	1	2	1	1	6	0

Table A.55: Overall Training Loss Results