CSCI447 - Assignment 4 - Design Document

Chris Major chrismichelmajor@hotmail.com
Farshina Nazrul-Shimim farshina287leo@gmail.com
Tysen Radovich radovich.tysen@gmail.com
Allen Simpson allen.simpson@motionencoding.com

Editor: (none)

1. Overview

The purpose of this document is to detail the design of three evolutionary algorithms to train a feedforward neural network. A genetic algorithm, differential evolution, and particle swarm optimization are to be implemented and compared to backpropogation as a means of training. The neural network will perform classification or regression on a series of six data sets, provided by the UCI Machine Learning repository (Dua and Graff (2017)). Experiments focused on the performance of the various training methods, with regards to convergence conditions, are performed.

2. Description

The assignment will be coded in F#, building on the previously constructed feedforward neural network and backpropogation features.

2.1 Experimental Design

The experiment will consist of four training trials for the feedforward neural network. The first will use backpropogation, serving as a control experiment and using existing code from the previous assignment. The second will use a genetic algorithm with real-valued chromosomes for training, the third will use differential evolution, and the fourth will use particle swarm optimization (PSO). Each trial will consist of three tests with 0, 1, and 2 hidden layers within the feedforward neural network.

Validation of the classification and regression conducted by the neural networks will be conducted via Mean Square Error (MSE) for all output nodes of the network. Determination of a correctly classified or regressed value will depend on the comparison between an observation's given class/regression value and the network-determined class/regression value. The results of these comparisons will be collected over various trials as defined by 10-fold cross-validation, then averaged to determine the algorithm's performance. Hypotheses regarding the terms of convergence, in the context of acceptable network performance, will be tested and analyzed in the official project document.

2.2 Major Decisions

2.2.1 Feedforward Neural Network

The feedforward neural network to be used is the same network developed for Assignment #3, based on Kulkarni and Harman (2011).

2.2.2 Backpropogation

The backpropogation functionality to be used is the same as was developed for Assignment #3, based on Rumelhart et al. (1985).

2.2.3 Genetic Algorithm

A genetic algorithm will be implemented as described by Booker et al. (1989), in order to train the feedforward neural network weights. A population of neural networks is taken from the connection matrix and set as a population. Two members are selected with replacement and evaluated based on "fitness" - how well they perform classification/regression. Crossover occurs to select the best components of each parent, resulting in two children to replace the parents of the population. Ideally, this process continues until the population demonstrates exceptional fitness and ability to correctly perform.

2.2.4 Differential Evolution

Differential evolution will will also be implemented as described by Storn and Price (1995), in order to train the feedforward neural network weights. Similar in nature to the genetic algorithm, this method focuses on real-value approximation. Three methods make up this algorithm: selection of the target vector (whether random or fitness-based), the number of difference vectors used to determine mutation, and the means of crossover used for creating the next generation of the population.

2.2.5 Particle Swarm Optimization

Particle Swarm Optimization (PSO) will also be implemented as described by Poli et al. (2007), for the purposes of training the feedforward neural network weights. The weights of our neural network will be represented as particles, "moving" towards optimal values based on personally-observed best values (`pBest`) and group-observed best values (`gBest`). PSO will be run for a set amount of time to allow for the particles to move towards their optimum values, with tuning parameters to place more weight upon personally-observed or group-observed bests. gBest tpology will be chosen to define which particles are permitted to share information.

2.3 Tuning

Words

3. UML Diagram

The UML diagram for this assignment is included below in Figure 1.

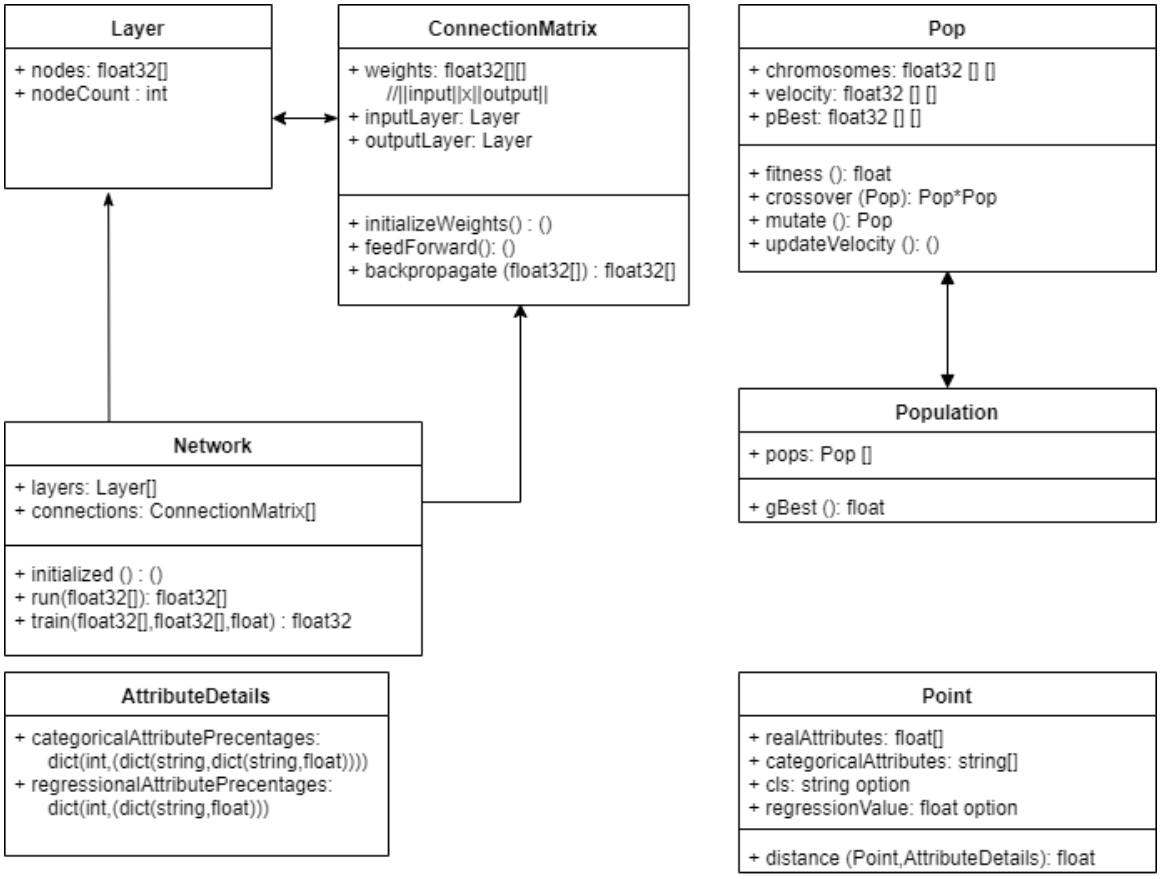

Figure 1: UML Diagram for Assignment #3

4. Class Description

4.0.1 Network

The Network class represents the feedforward neural network, containing an array of layers and a matrix representing the connections. initialize() randomizes the weights of the matrix. run() takes an input array and returns the resulting output array. train() takes the input array, the expected output array, and the epsilon (accepted change in error) as parameters and runs the feedforward neural network until convergence, then returns the error.

4.0.2 Layer

The Layer class represents a layer - input, output, or hidden - within a Network. It has an array of nodes, represented by floats, and a nodeCount value representing the number of nodes in that layer.

4.0.3 Connection Matrix

The ConnectionMatrix class represents the connections within the Network. A 2D array of weights, represented by floats, an inputLayer, and an outputLayer make up this class. initializeWeights() randomly assigns initial weights to the ConnectionMatrix. feedForward() takes the input layer and cross-multiplies its nodes with the weights. backpropogate() is the original backpropogation functionality from the previous assignment and simply performs backpropogation by taking the delta values for the outputLayer and using them to modify the weights as need be.

4.0.4 Population

The Population class represents the general population of an evolutionary or population-based algorithm. It is made up of an array of Pops, with a function returning the best of the population through gBest().

4.0.5 Pop

The Pop class represents an individual within the class. It has a 2D array of chromosomes, a 2D array of velocities, and a 2D array of pBest "personal best" values. The fitness() function builds a neural network with weights associated with the chromosomes and runs the neural network forward, returning the accuracy of the answer. The crossover() function takes one Pop and crosses it over with another Pop, then returns the children. mutate() applies mutation to the chromosomes of the current Pop. updateVelocity() updates the velocity of the current Pop.

4.0.6 AttributeDetails and Point

AttributeDetails is a structure used to store the calculations required to find the distance between two points. A Point is an object that stores the real attributes, categorical attributes, and the class/regression value of each point fed into the network.

5. Summary

Per these design guidelines, we aim to deliver three evolutionary algorithms and an analysis of their effects on the training of a feedforward neural network. Upon implementation and tuning, experiments will be conducted on provided data sets to determine metrics of performance and convergence. A formal report detailing the results and analysis will follow upon completion of the project.

References

L.B. Booker, D.E. Goldberg, and J.H. Holland. Classifier systems and genetic algorithms. *Artificial Intelligence*, 40(1):235 – 282, 1989. ISSN 0004-3702. doi: https://doi.org/10.1016/0004-3702(89)90050-7. URL `http://www.sciencedirect.com/science/article/pii/0004370289900507`.

Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. URL `http://archive.ics.uci.edu/ml`.

Sanjeev Kulkarni and Gilbert Harman. An elementary introduction to statistical learning theory. 04 2011. doi: 10.1002/9781118023471.

Riccardo Poli, James Kennedy, and Tim Blackwell. Particle swarm optimization: An overview. *Swarm Intelligence*, 1, 10 2007. doi: 10.1007/s11721-007-0002-0.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

Rainer Storn and Kenneth Price. Differential evolution: A simple and efficient adaptive scheme for global optimization over continuous spaces. *Journal of Global Optimization*, 23, 01 1995.