CSCI447 - Assignment 3 - 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 two neural networks - a feedforward network with backpropagation, as described by Rumelhart et al. (1985), and a radial basis function network, as described by Moody and Darken (1989). These algorithms are performed 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 algorithm, with regards to convergence conditions, are performed.

2. UML Diagram

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

3. Description

The assignment will be coded in F#, with the use of SIMD intrinsics to accelerate matrix multiplications. As the last assignment struggled with excessive time complexity issues, we aim to reduce the time spent on matrix operations using intrinsic operations from the .NET development environment.

3.1 Experimental Design

The experiment will consist of three trials for the feedforward network and four for the radial basis function network. For the feedforward network, individual trials will be run based on the number of hidden layers - 0, 1, and 2, with backpropogation implemented for each trial. For the radial basis function network, the previous assignment's algorithms will be used to feed reduced datasets. Specifically, the Edited Nearest Neighbor, Condensed Nearest Neighbor, K-Means Clustering, and K-Medoids Clustering output data sets will be fed into the network as inputs.

An update in the project's requirements allows for the use of either ENN or CNN, depending on which produces the smaller data set.

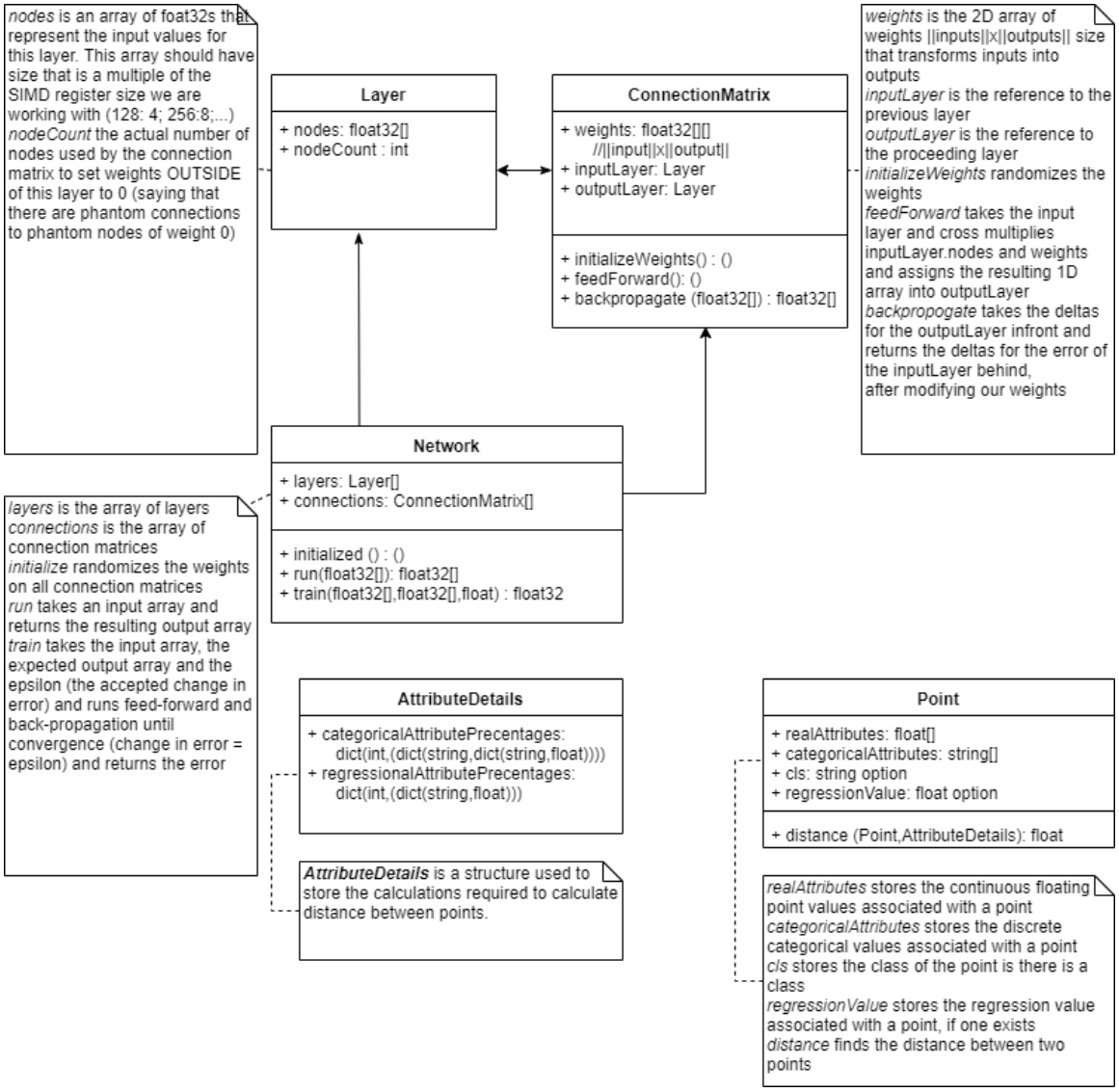

Figure 1: UML Diagram for Assignment #3

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.

3.2 Major Decisions

3.2.1 Feedforward Neural Network

The most significant choice made regarding the feedforward neural network is the activation function, as described by Kulkarni and Harman (2011). For the purposes of our experiment, a sigmoidal function was chosen - specifically the logistic function. This function and its derivative are described as follows:

$$y = \frac{1}{1 + e^{-x}}$$

$$y' = y(1 - y) = \left(\frac{1}{1 + e^{-x}}\right) - \left(\frac{1}{1 + e^{-x}}\right)^2$$

As the logistic function allows the outputs of the network to be described as probabilities, we have elected to use this as our activation function to ensure for easier interpretation of our experimental results. In the case where time-saving intrinsics are implemented during the experiment, the Taylor Series approximation of y and y' may be used to meet the demands of such low-level instructions.

Another design choice concerns momentum, as described by Sarigül and Avci (2017). Momentum is the ability for the neural network to push the process away from local minima by referring to previously calculated weight values. This feature is implemented within the feedforward neural network's function call, allowing the user to turn momentum on or off for a given experiment. The equation for a momentum-affected weight Δw_{ji} is as follows:

$$\Delta w_{ji}^t = -\eta \left(\frac{\delta Err}{\delta w_{ji}}\right) + \alpha \Delta w_{ji}^{t-1}$$

Here, t represents the current set of weights being calculated and $t - 1$ refers to the previous set of weights previously calculated.

3.2.2 Radial Basis Function Neural Network

For the Radial Basis Function (RBF) neural network, a Gaussian basis function will be used as the activation function for the hidden layer nodes. The receptor for each node of the hidden layer will be the center of each cluster obtained from the previous experiment. The output layer will be obtained using a linear activation function with the hidden layer nodes, per the description of Broomhead and Lowe (1988).

RBF neural networks overcome the limitations of linear inseparability of the input dataset as a single hidden layer projects the input into higher dimension with the help of the Kernel functions. In this case, the choice of a Gaussian basis function plays an important role in the output of the network as it is symmetric or radial from the center, and the function decreases proportionally with the distance. Additionally, because of its finite response, this function is biologically plausible (Orr (1996)).

The Gaussian basis function is expressed as:

$$K_g(x_i, x_j) = e^{-\frac{(x_i - x_j)^2}{2\sigma^2}}$$

Here, selection of standard deviation σ and receptor x_j determines the shape of the function. A significantly small σ can result in over-fitting and vice-versa.

3.3 Tuning

3.3.1 Feedforward Neural Network

Three parameters are to be tuned for the feedforward neural network. The first is η, a value to reduce the error of the new weight computed by the momentum equation. The second is α, a penalty value assigned to the previous weight of the optional momentum component. These values will be tuned manually as the experiments are conducted.

Another parameter to be manually considered is the number of nodes per layer. This is to be established as an input parameter for the feedforward neural network's function calls.

3.3.2 Radial Basis Function Neural Network

As a part of the Gaussian function, the σ values are to be tuned, and as a part of the output nodes, the output weights require tuning. These can be done manually or through various statistical functions - the turning method to be used will depend on the progress of programming as the deadline approaches.

4. Summary

Per these design guidelines, we aim to deliver two operational neural networks to satisfy the assignment requirements. 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

D.S. Broomhead and D. Lowe. Multivariable functional interpolation and adaptive networks. *Complex Systems*, 2:321–355, 1988.

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.

John Moody and Christian J Darken. Fast learning in networks of locally-tuned processing units. *Neural computation*, 1(2):281–294, 1989.

Mark J. L. Orr. Introduction to radial basis function networks, 1996.

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.

M. Sarigül and M. Avci. Performance comparision of different momentum techniques on deep reinforcement learning. In *2017 IEEE International Conference on INnovations in Intelligent SysTems and Applications (INISTA)*, pages 302–306, July 2017. doi: 10.1109/INISTA.2017.8001175.