Design Document: Function Approximation with Neural Networks

Amy Peerlinck, Na'Shea Wiesner, Scott Martin, Taylor Heinecke

October 2, 2017

1 Description of the Project

The main goal of the project is to perform function approximation. Function approximation is a problem that consists of constructing a model that implements a certain function $f : X \to R$. However, the actual function is never learned exactly, it is simply approximated by the model, thus the name of the problem. The functions to be approximated in this project are five different versions of the Rosenbrock function [4], which is defined as follows:

$$f(x) = f(x_1; x_2; ...; x_n) = n_1 X_i = \sum_{i=1}^{n-1} [(1 - x_i)^2 + 100(x_{i+1} - x_i^2)^2] \tag{1}$$

The model is represented by two different types of Neural Networks (NN), which are learned to approximate the aforementioned function. The first is a Feedforward Neural Network with Backpropagation (FFBP NN) and the second is a Radial Basis Function Neural Network (RBF NN). The rest of the document is structured as follows. Section 2 explains the Software Architecture by means of a UML diagram. The Design Decisions of this architecture are detailed in Section 3. The Experimental Design for each of the NNs is outlined in Section 4, along with an the assertion of the hypothesis and the evaluation metrics for each algorithm.

2 Software Architecture

In Figure 1, a UML diagram is shown outlining the architecture of the Neural Networks. The UML diagram consists of the interface class Neural Network, from which the FFBP NN and the RBF NN extend, Neuron and Connection classes and a K-MeansCluster class (that will be implemented by the RBF NN) utilizing the Cluster and Datapoint classes. The abstract NeuralNetwork class contains the *initialize()*, *initializeWeights()*, *train()*, and *test()* methods, because these are paramount to every neural network. Furthermore, it dictates that every NN has to consist of an input layer and an output layer and that it should store the amount of nodes each of these layers has. Each NN also consists of *Neurons* and *Connections* between those neurons. Each *Neuron* has its own weight and the possibility of a *sigma* value (which can be null). The Feedforward Backpropagation NN initializes its hidden layers based on the desired amount of hidden layers (0, 1 or 2). This network also contains specific hyperparameters and the necessary functions to train the data. The Radial Basis Function NN always contains one hidden layer, the type of function to be used and calls the K-Means Clustering algorithm to initialize its inputs. [4]

3 Design Decisions

When considering how this project could best be structured, the central concept of neural networks was kept in mind. Because Neural Networks are complex structures that perform several different calculations, an interface class was chosen to determine the essential properties and functions of a NN. This ensures that each NN extending from this class contains the same basic structure of an input and output layer and that it implements training and testing methods. The intricacy of a NN also led to the design choice of keeping all its functions within its own class. Assigning these functions to other classes would result in an overabundance of classes and create a messy class structure. In order to slightly lessen the load on the NN implementations, Neuron and Connection classes are created, thus storing information on the model structure. The K-Means Clustering algorithm is a separate instance because it is an algorithm that stands by itself and is simply used to initialize the RBF NN. This follows the concept of separation of concerns because K-Means Clustering is intended to cluster the input data and then hand it off to the RBF NN, which “transforms” the input data, and is mostly concerned with the output results.

4 Experimental Design

In this section the experimental design of both NN implementations will be discussed separately. After analyzing the design of these NNs, a hypothesis is stated and lastly the evaluation metrics to compare the results are briefly explained.

4.1 Multi-layer Feedforward Network with Backpropagation

The Multi-Layer Feedforward NN with Backpropagation has several parameters that need to be chosen before the network can be initialized. These include the number of inputs, hidden layers (0,1 or 2), the amount of hidden neurons in those layers and outputs (who are reused as inputs) for each run. To adjust these parameters, gradient decent can be used to ensure optimal performance.

Furthermore, for each neuron it must be decided whether a linear or sigmoidal activation function will be used. Whether or not momentum is used during the learning process, is also to be decided. The momentum factor influences the calculated weight change, to prevent the weight itself from oscillating. [1] [2]

4.2 Radial Basis Function Neural Network

A Radial Basis Function Neural Network is an artifical, feedforward neural network that can be used for function approxiamtion using three, and only three layers. The activation functions for RBF NNs are known as radial basis functions and at the output layer, a linear summation is applied on the hidden layer's outputs and weights.

In this implementation, the RBF NN will take in an arbitrary number of inputs, Gaussian basis functions, and outputs. Three different numbers of Gaussian basis functions (3,5, and 7) will be tested for comparison on the Rosenbrock function for $n = 2; 3; 4; 5; 6$. [3]

4.3 Hypothesis

Based on brief research into the algorithms and the functions, the FFBP NN with 2 hidden layers and the RBF NN are expected to perform similarly as far as approximation is concerned. However, the FFBP NN will probably converge much slower than the RBF NN, especially if the initial learning rate is poorly chosen. It is predicted that the RBF NN will more accurately approximate the Rosenbrock function when the number of radial basis nodes increases. However, the cost associated with the number of radial basis nodes will increase as well. It is also predicted that as the number of hidden layers and the size of the hidden layers increase, the approximation accuracy will go down while the rate of convergence goes up.

4.4 Evaluation Metrics

When evaluating the Multi-layer Feedforward Network with Backpropagation, the squared loss will be calculated at the output layer by a summation of the delta errors of all the output nodes and then compared to the expected output. This process is given by the following equation:

$$Err(x) = \frac{1}{2}\sum_{x}^{X}(d_x - o_x)^2 \tag{2}$$

If the error resides outside of the threshold, Backpropagation is necessary and the weights of the network will be updated using gradient decent on the delta errors.

In order to evaluate the Radial Basis Function Neural Network, the squared error is calculated at the output layer and compared against the error threshold.

If the error is too large, gradient decent is used to update the weights until the error is within the threshold. [5]

References

[1] G Bebis and M Georgiopoulos. Feed-forward neural networks.

[2] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators, Mar 1989.

[3] J Park and I Sandberg. Universal approximation using radial-basis-function networks, Jan 1991.

[4] John Sheppard. Machine learning and soft computing lecture notes, Fall 2017.

[5] J.M. Twomey and A.E. Smith. Performance measures, consistency, and power for artificial neural network models, Jan 2000.

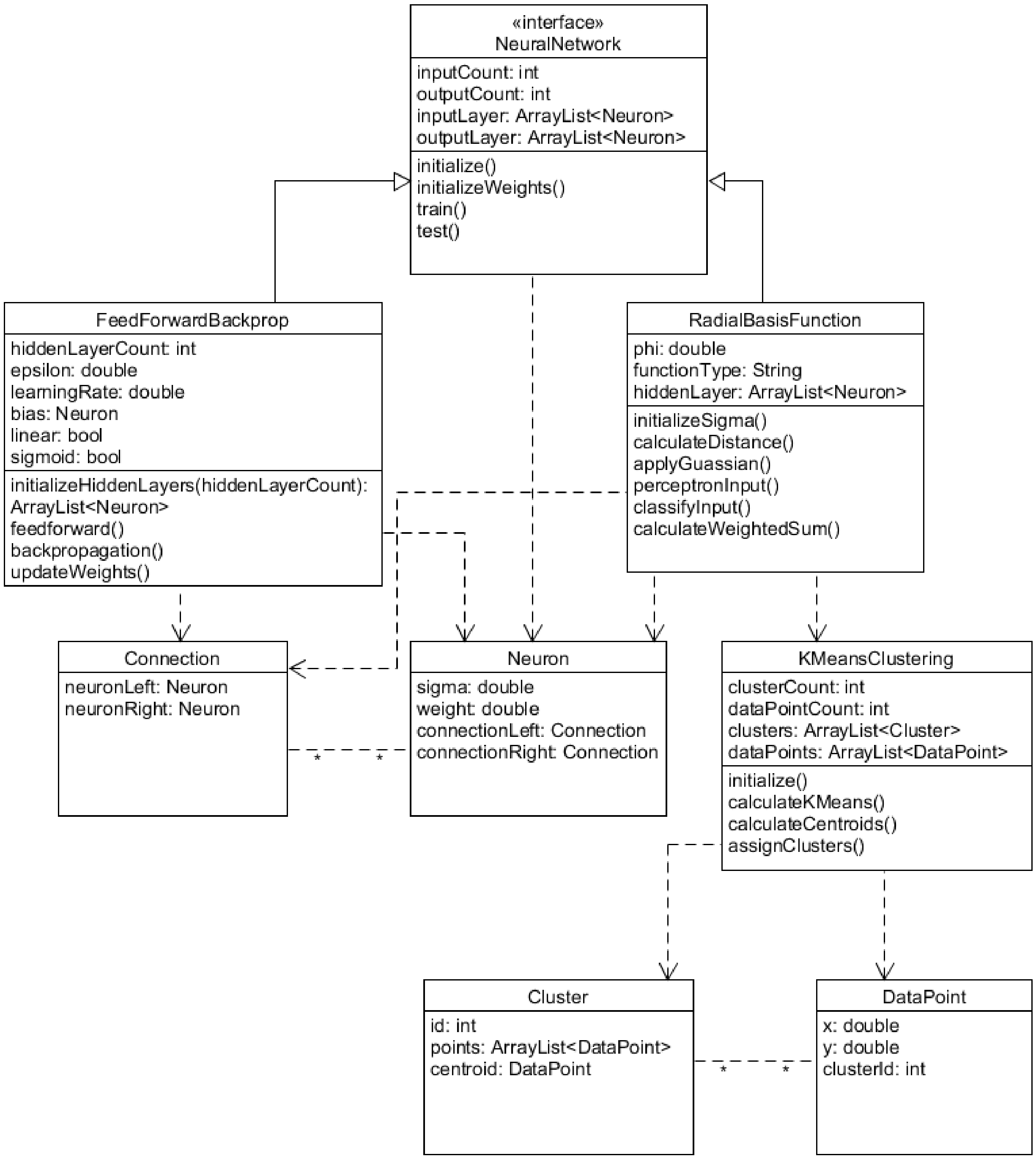

Figure 1: UML Diagram of software architecture..