Preston Percival, Richard Padilla, and Joel Mwesigwa

CSCE 315-100

June 8, 2017

TEAM 6 Design Document:

Section 1 – Introduction to the Project

For project two our group's goal is to design a program that can recognize a capital letter input into a 5x7 grid using a neural network. To accomplish this, we are dividing our project into three main parts. The three components of this project are the GUI which will be used by the user to manipulate the input, the neural network which will recognize the letter based on past trials, and finally the back propagation which will be used to train the neural net to be more accurate. Once we design all three discrete parts we will put them together to form a cohesive program that can be used by anyone.

Section 2 – High Level Entities

Neural Network: Athena is a Neural Network designed to detect any of the standard 26 Uppercase letters of the English Alphabet on a 5x7 matrix. Athena is designed to work with our back propagation algorithm to learn on a 5x7 dot matrix font how to recognize letters, even with some noise. Once trained, Athena will be run in conjunction with our GUI, and attempt to discern letters on a 5x7 grid that a human user can change in real time.

GUI: The GUI will be programmed through JavaFX and consist of a main screen with a 5x7 grid of buttons that when clicked will change colors and represent a boolean 0 or 1. This screen will also have two extra buttons below the grid. These buttons will be the evaluate button which will activate the Neural Network and then there will also be a reset button which will reset the grid. We will also have a second screen that appears after evaluating that will tell what letter the Neural Network thinks it is and perhaps the coefficients associated with possible other letters.

Back Propagation: The back propagation function will return a neural network and will take examples each with and input and output vector, a layer with several levels, weights and a sigmoid function that we implemented in the first project. It will also have a vector of errors indexed by a network node.

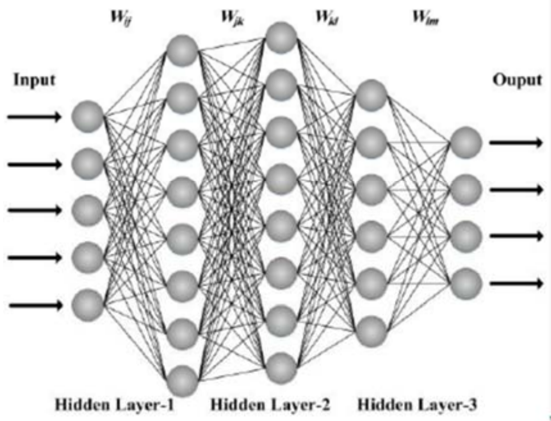
Section 3 – Low Level Design

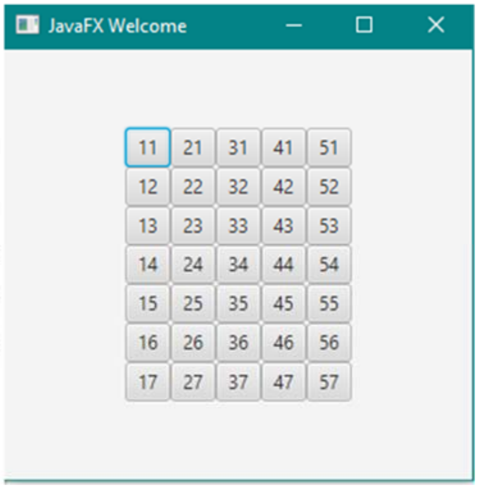
Neural Network: Athena is a Neural Network composed of layers, which in turn are composed of neurons. The first layer, called the perceptron, takes in input from the 35 squares of the 5x7 matrix. Each neuron in the first layer will receive the 35 inputs, as well as a threshold, and compute a value based on the weights, inputs, and threshold. Once each neuron has processed the input, the output will be sent as input to the next layer. After the last layer has processed the input, the output will be sent to an array consisting of 26 spaces. Each element will correspond to a letter, and the number represents the percentage that Athena believes the input is that letter. During training, the output will be sent to the back-propagation algorithm, in order to strengthen Athena's accuracy. After training, the output will display the top 5 letters that Athena believes the user has made.

GUI: The largest part of the GUI will be the individual buttons in the 5x7 matrix, each of which will have several traits associated with it. At the start, all 35 buttons will be white, which is associated with a boolean 0, but once pressed the color of the button will be changed to black representing a 1. These will feed into the neural network and be the values for which it evaluates when the "Evaluate" button under the grid is clicked. The "Reset" button next to the evaluate button will change all of the grid buttons back to the color white and reset their boolean value to 0. Once you are ready to use the neural network to identify your character the evaluate button will be pressed and then it will move to a new screen that list the top 5 guesses for the character and the statistics associated with that decision.

Back Propagation: Back propagation will determine the correct output prediction based on the inputs provided. Each one of the inputs has a weight associated to it. These weights are stored in a vector. We use these weights to compute the values of the hidden layers which are also stored different vector we use the dot product of the vectors to archive this. Since the weights are fixed we can easily compute the hidden weights at each level (layer). We do this by running the sigmoid function. We keep doing this layer by layer until we get to the end of neuron obtaining the desired output which is our prediction. With this prediction and a true value, we can compute the error which is how far we are from the desired output. From the error, we determine whether the output should be higher or lower so we use that information to determine if the layer from the previous hidden layer should be higher or lower, then that determines if the other previous layer should be higher or lower and so on. This is the hidden training in the back ground which would be done again to get the closes correct output.

Figure 1 – Neural Net **Figure 2 – Primitive GUI**





<u>Section 4 – Benefits, assumptions, risks/issues</u>

Undertaking this project, we have several assumptions. First of all, we assume our sigmoid function is efficient, we assume that a neural net can be trained through back propagation, and we assume that the input is confined to a 5x7 grid. Being confined to this grid provides the benefit that we will have a finite amount of possibilities for input, while training a neural net to recognize these patterns provides the benefit that we do not have to explicitly state the exact output for each input. By splitting the project into three discrete parts we also have the benefit of being able to develop in parallel and therefore be more efficient.

Assumptions:

- the sigmoid function is efficient
- the neural network can be trained through back propagation
- input is confined to 5x7 grid

Benefits:

- finite possibilities for input
- we do not have to explicitly state the exact output for each input
- splitting the project up allows our group to be more efficient
- Plain and simple GUI

- adjustable learning rate for the back propagation
- deep learning neural network with adjustable sigmoid function

Risks/Issues:

- back propagation might overcorrect and make the system too rigid
- back propagation might not teach the neural network enough and lead to inaccuracies