# MSDS-458 Assignment #1

# Classifier NN's

Mark McGown

## ❖ Introduction

The goal of this assignment was to explore how classifier neural networks work by designing, training, assessing code pre-prepared by AJM (Maren, Guenther, 2019) on a set of binary lists representing letters. In an iterative manner, changing the inputs dramatically altered the output. This was surprising considering several of the inputs are assigned randomly and converge upon a final solution that is commonly very similar for a given configuration. More so, input parameters that also affect this convergence were iteratively changed. The visual heatmap in pixel form representing this state of convergence will be provided and discussed as they relate to the function of a classifier neural network. From that, the function and purpose of hidden layers in a NN will have conclusions drawn.

## ❖ Data Preparation & Design

The input to this classifier neural network was prepared by AJM to be Python list representations of 9x9 binary images of letters of the first 16 letters of the alphabet. 0's & 1's were used in each value of the list to represent, when turned into an array, if the letter of the alphabet occupied that specific cell. An example of what this array would look like is shown in Figure A below. Certain letters were also contributed by previous students of the MSDS458 class. The data type given to the NN for each letter was a tuple and shown in Figure B, as described in the reading material from Week 3.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

*Figure A: 9x9 Binary Image of the Letter "A"*

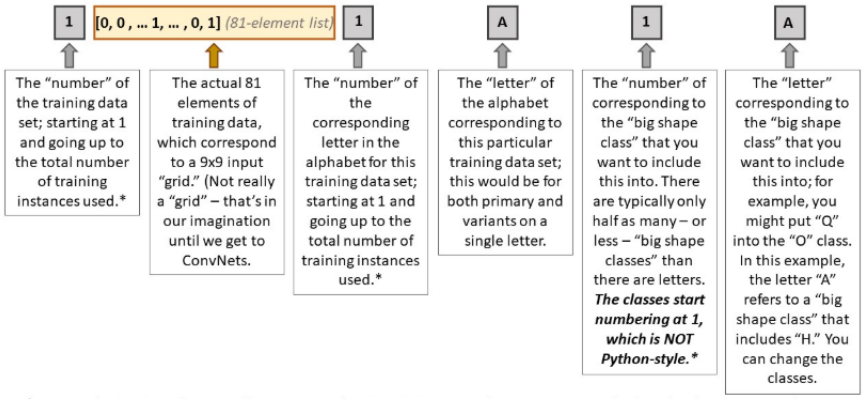


*Figure B: Block Structure Diagram of Training Set Contents (Maren, 2019)*

Besides the training data itself, data used in this analysis included the neural net layer size specifications, parameter definitions for backpropagation, and the random weights assigned between layers as well as for the biases in each layer. The neural net layer size specifications were fixed with an input size of 81 and an output size of 9 whereas the size of the hidden layer was manipulated iteratively. Parameter definitions for backpropagation included alpha, eta, max number of iterations, epsilon, max number of iterations, SSE, and the number of training data sets (fixed for this experiment).

Here, alpha represents the non-linearity of the sigmoid function, eta is the training rate parameter (scales amount of change to weight), epsilon is the convergence of SSE needed to finish training, max number of iterations is how long the program should pursue this epsilon convergence, and SSE is the sum of squared error of the neural net predictions vs reality. Finally, the number of hidden layer nodes represents the level of complexity given to the final neural net. Often having too few nodes can result in the underfitting if the problem is very complex. Conversely, having too many nodes for a problem runs the risk of overfitting the

complexity. The balance between these two opposing forces must be struck in order to best estimate what configuration achieves the lowest SSE possible.

## ❖ Exploration & Results

For the purpose of comparison, the model was first run at the default configuration settings as shown in Figure C. It can be seen that “I” and similar letters that contact a similar vertical line shape are warmer in the heatmap for Node 0 whereas rounder letters have little activation for this node. Node 3 seems more activated by letters like “H” with a horizontal line connecting what might resemble two vertical lines -notice how letters like “A” and “P” also partially activate this node since they partially meet this criterion. Overall, the neural net’s hidden layer activations align with what one might visually perceive in the similarities of these letters. For example, one might perceive “H” to be like the letter “A” with the aforementioned Node 3 attribute of the horizontal connection but with a slight difference that Node 4, with its lack of activation for “A,” might be picking up on.

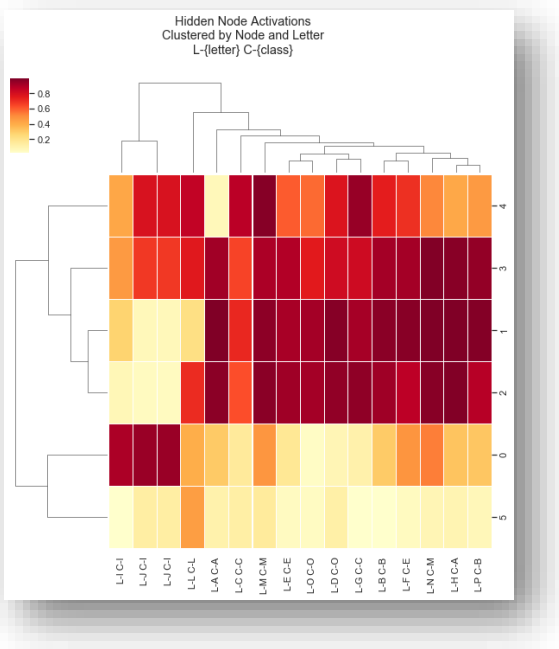


*Figure C: Clustering of Hidden Layer Activations (Default)*

The model was then run on a hidden layer size of 1 node and the results are shown below as a heatmap representation of the final activation values for each letter and class in Figure D. Letter ‘B’ and similarly shaped letters such as “E” show relatively low activation values whereas letters tending towards more diagonal lines such as “N” and “M” show a very high activation for this single node. Interestingly, the gradient in activations is not necessarily linear, but upon further inspection the change in shape between letters is often very drastic, too. In some iterations of running with this size, it found “B” dramatically different than “M” & “N” whereas other times it found “I” dramatically different than “M” & “N”. It is also interesting to note that Figure D does not appear to represent Figure C’s average-per-letter activation across the prior nodes. Partly this is surely due to the random starting weights converging on any minimum change in SSE they can achieve. That can be seen by re-running the model with any

configuration fixed and observing drastically different heatmap results. However, what is common so far is that very similar letters such as “I” + “J” and “M” + “N” are always at relatively similar activations -no matter what the actual activation is or what node it is on.

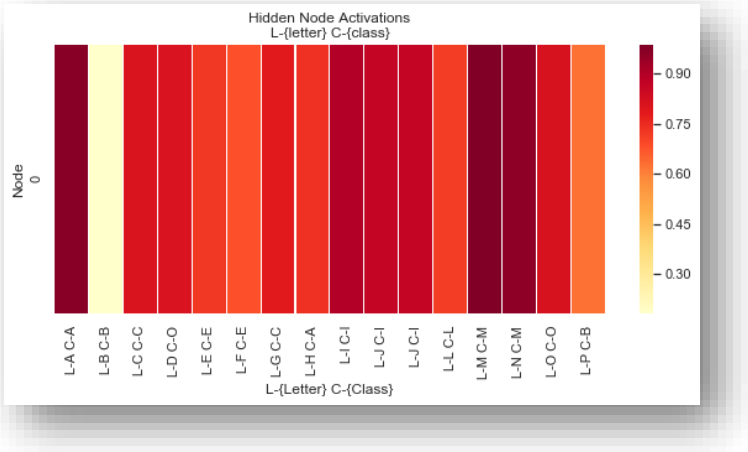


*Figure D: Single Hidden Layer Node Activations*

Setting the number of nodes back to 6, the transfer function’s shape and its affect on the hidden nodes was explored by modifying alpha. At the original default value of 1 for alpha, the gradient across all cells in the above heatmaps are very much not linear. Most cells in Figure C specifically show as being closer to 0.2 or 0.8 as their activations rather than in-between. This is because an alpha of 1 results in the transfer function being the classic sigmoid function as shown in blue in Figure E below. Modifying alpha to a value of 0.1 results in a transfer function that is much more linear as shown in red in Figure E.

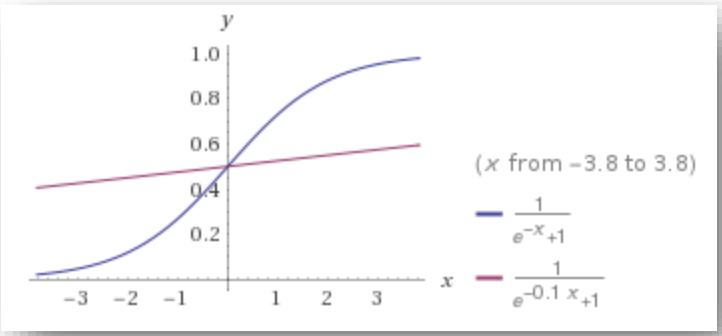


*Figure E: Sigmoid Function vs. Modified Alpha Sigmoid Function*

Re-running the default model settings with only alpha adjusted from 1 to 0.1 shows a more reduced gradient in activations in Figure F. Though the default settings could sometimes achieve an SSE of 0.02 in some cases, iterations of this new model with modified alpha showed a decrease performance in overall SSE. Letter clustering in Figure F vs. Figure C shows the less definite purpose of each node made the relations between clusters less clear. Since transfer functions convert weights, x, into activations, y, a poorly chosen transfer function may be the result of allowing too much/too little ambiguity in the roles of each node. To contrast, the danger of a step-wise transfer function might be the complete maximization/minimization of activation in weights that were still close to their initial randomly assigned values -thereby giving little to no room for weight refinement in the backpropagation process. In a certain way, the “step” in transfer functions that are more step-wise is too hard to overcome for backpropagation. In a similar way, a linear transfer function allows too much refinement and appears to also run the risk of early convergence when two subsequent iterations are similarly ambiguous enough to achieve a SSE delta less than epsilon.

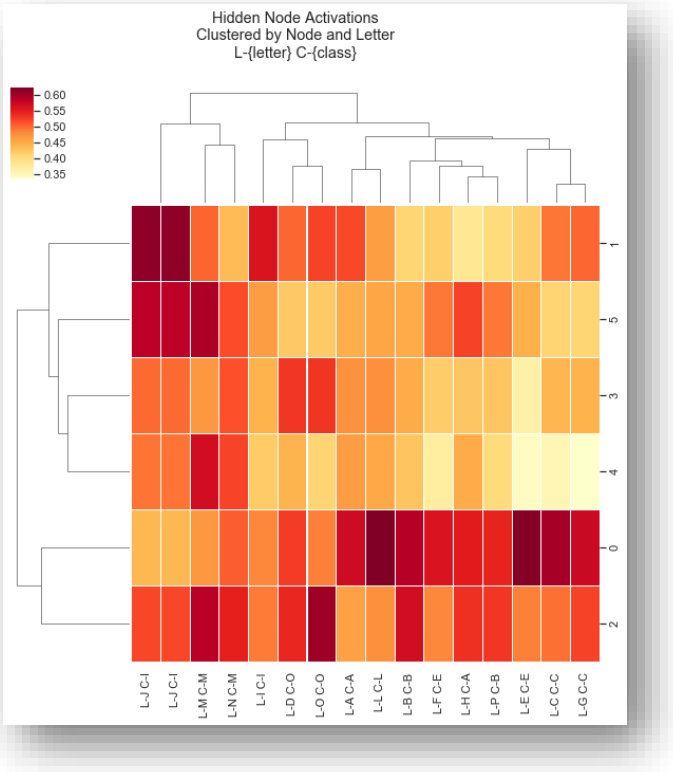


*Figure F: Hidden Node Activations with Reduced Alpha*

The training rate parameter, eta, was modified next with some interesting results as shown in Figure G. Due to being back to the sigmoid function as the transfer function in default, the contrast in activations is starker. Node 3 in this iteration had not been used and the SSE converged to 0.3 instead of a more global minimum in these experiments of an order of magnitude less. Why might this node have been excluded? With a slower learning rate, it appears a local minimum regarding epsilon has occurred. In other words, by adjusting weights less so than previously, SSE change was found to be insignificant for more backpropagation to continue & attempt more values along the sigmoid function for Node 3. Iterating this parameter seemed to show that, although alpha could change the range in the landscape of activations possible (how stark/how similar), training rate was the potential for the program to

continue exploring the landscape in activation values once it found suitable local minimum curvature for acceptable epsilon. A value too high for eta could theoretically force convergence by exceeding the more fine-grained backpropagation necessary for the most (and maybe nearly attained) optimal NN. In Figure G, an entire aspect of complexity was not found in Node 3 and thus SSE was impacted an order of magnitude. This resulted in the most similar letters being still clustered together but less understanding of how the clusters of those letters related for the more in-between features of letters.

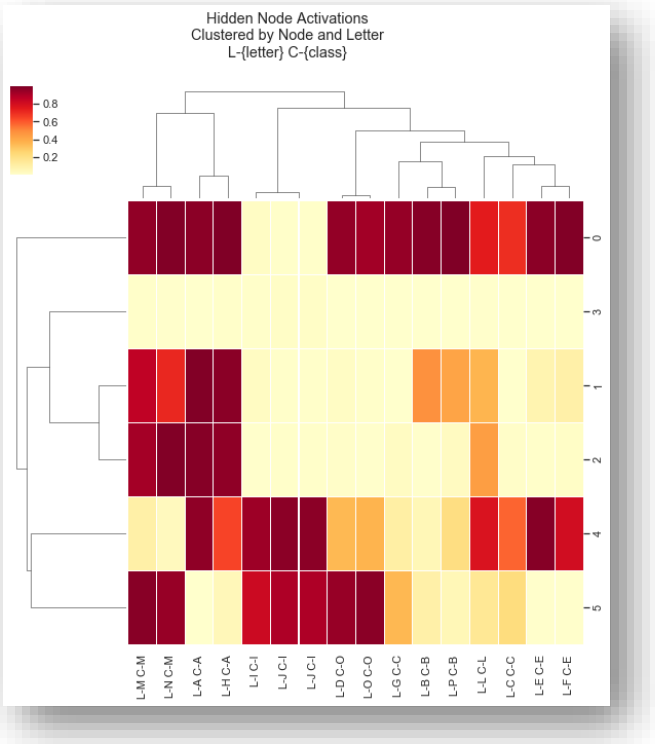


*Figure G: Hidden Node Activations with Reduced Learning Rate*

## ❖ Conclusion

The results of this experiment found evidence for a fine balance that must be struck in designing a neural net across several parameters simultaneously. With so many moving parts,

this experiment underscores the importance of not blindly entering this design space; rather, the relative complexity of classification, precision in finding the optimum in this complexity, and distribution of activations (via transfer function) must be first at least somewhat understood.

What can be concluded about the purpose and function of hidden layers in a NN? As stated previously, we know that misconfiguration in hidden layer size can result in over-fitting/under-fitting, and we exit this experiment with some idea of the landscape they help enable w/r/t input parameters. In the previously studied X-OR problem, we had a much smaller size of hidden layer nodes clearly due to the smaller number of input parameters combined with the reduced possible outputs those inputs might represent. In this way, the hidden layers represent the landscape of the design space we constrain the parameters to operate in - somewhat like degrees of freedom. Many of the inputs of course constrain this design space further, like when a reduction in alpha reduced their significant differences in activation per letter, but even in such experiments still retain the character of the maximum space that can be navigated.

As always, alternate approaches exist. Dropout is one such method that prevents more hidden nodes from overfitting while still allowing them to spend more time individually focusing on their place in the overall reduction of SSE. Another such example might include a learning rate that increases non-linearly over the number of iterations (Zulkifli 2018). Such automated processes may also store the local minima found for later iteration reference while proceeding to ensure the validity of that finding. As we will see in later experiments, hidden nodes and the parameters that affect them are the primary characters in the story that the narrative plays upon.

## References

Maren, A. J., & Guenther, R. (2019, April 18). 81-Input-Node Classifier (Many Output Classes). Retrieved October 4, 2019, from https://canvas.northwestern.edu/courses/101750/files/7348174/download?wrap=1.

Zulkifli, H. (2018, January 27). Understanding Learning Rates and How It Improves Performance in Deep Learning. Retrieved October 4, 2019, from https://towardsdatascience.com/understanding-learning-rates-and-how-it-improves-performance-in-deep-learning-d0d4059c1c10.