2019SP_MSDS_458-DL_SEC56 Week3- Assignment1

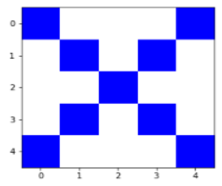
Prabhat Thakur

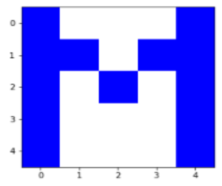
25-Input-Node Alphabet Letter Classification (5 Classes)

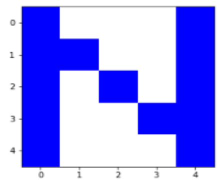
The goal of this assignment is to understand how neurons/nodes in simple neural network hidden layer learns to represent features within the input data. The data used in the assignment is a collection of letters from the English alphabet in the form of arrays. Through this research, it is possible to understand and interpret the functioning of hidden layer nodes and back propagation algorithm as a way to ultimately generate the correct output. The challenge of this assignment is less about training a successful neural network for deployment but to understand and interpret the hidden node activations as features.

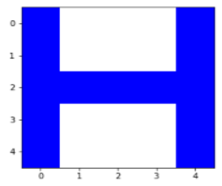
To research how hidden nodes represents data features, the input data used in this analysis is a small collection of 5 English alphabets letters "X", "M", "N", "H", and "A" . Each letter is represented by 5 by 5 pixel size. There is only one representation/sample of each letter considered in the analysis. This matches the intent of the analysis not being about predictive power on out-of-sample data, but rather to interpret hidden layer results. The Python notebook script, “MSDS458Week3Assighment1.ipynb”, visualizes each of the letters individually. Examples are shown in Figure 1. All images are of same 5x5 pixel size, this consistency is important to the kinds of features or rules that may be represented.

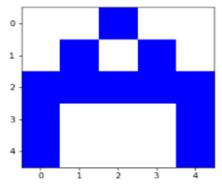
Figure 1 - Alphabets representations used for training the neural network











The core building block of neural networks is the layer, a data-processing module that works as a filter for data. Some data goes in, and it comes out in a more useful form. The layers extract representations out of the data fed into them, representations that are more meaningful for the problem at hand. Most of deep learning consists of chaining together simple layers that will implement a form of progressive data distillation. A deep-learning model is like a sieve for data processing, made of a succession of increasingly refined data filters—the layers. The architecture of this "character recognition network" consists of a sequence of two Dense layers, which are densely connected (also called fully connected) neural layers. The first layer also called hidden layer consists of 8 eight nodes and the second (and last) also called output layer is a 5-way softmax layer, which means it will return an array of 5 probability scores (summing to 1). Each score will be the probability that the current alphabet image belongs to one of the 5 alphabet classes. Python keras package is used to train the network and following are the parameters:

```
#Neural Network with Densely connected (fully connected) Single hidden layer
    network = models.Sequential()
    network.add(layers.Dense(8, activation='relu', input_shape= (25,)))
    network.add(layers.Dense(5, activation='softmax'))

    network.compile(optimizer='rmsprop',
    loss='categorical_crossentropy',
    metrics=['accuracy'])

    network.fit(training_data, training_labels, epochs=500, batch_size=5)
```

Hidden nodes "generalize" input data, they detect combinations of active nodes and can be referred as "feature detection." Thus, the way in which it can be reverse-engineer is what a given NN is doing to identify what the hidden node activations are in response to a given input pattern, and to compare hidden node activations across the sets of inputs (and of course, their desired outputs). To do this, once neural network is trained, hidden node activations are analyzed for each different (kind of) input. Start looking for hidden nodes that might possibly be active for a given feature-type (diagonal, vertical bar, etc.). When looked at these letters from perspective of having a 2-D visual cortex, following features can be identified to distinguishing 5 different characters:

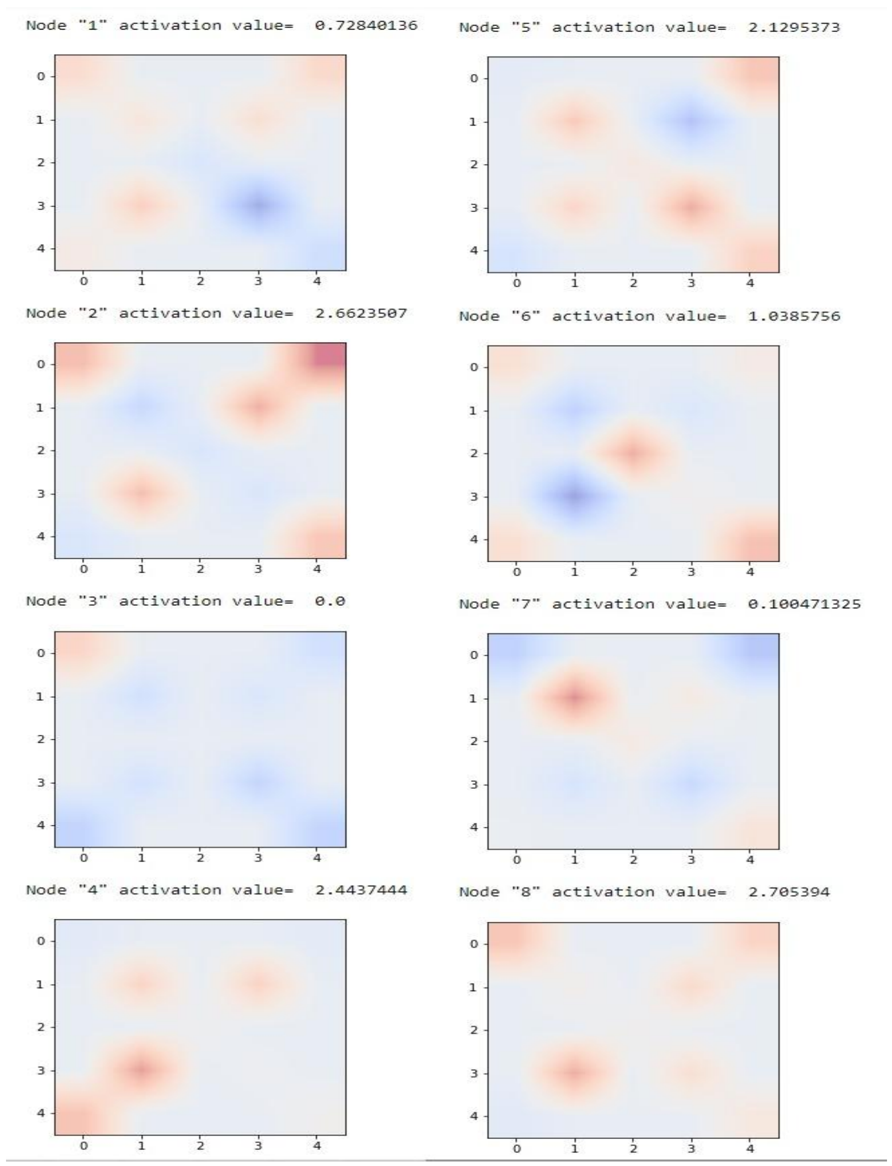
- X - two crossing diagonals
- M - two partial diagonals, two vertical bars at L & R-hand sides
- N - this should be tricky - and possibly get caught between 'X' and 'M' - has one of the diagonals, and both of the vertical bars at L & R-handsides
- H - same two vertical bars, plus a cross-bar

- A - again, possibly tricky - the vertical bars are shortened, it is the only character in this set with some nodes in the top-middle area, and it has the horizontal bar (like the letter 'H');

The first step of hidden node output interpretation is to visualize the weight matrix from the input layer to the hidden layer for each hidden layer node. The red areas of the visuals indicate parts of the matrix where an input of one will increase the activation while blue areas decrease the activation for a non-zero input. It may be easy to focus interpretation efforts on the positive weights but the negative weights are equally important in analyzing the results. For each node, it is also important to view the product of a single letter input with the weight matrix. Visualizing that product shows which weights are active which is helpful in understand what is leading to whether that node is activated for a specific letter. By repeating that analysis for multiple letters, it is possible to see which letters are interacting with the hidden layer in similar ways.

Node 1 appears to get activate for letters with diagonal lines like "X", "M" and "N" and the intersection of horizontal line and right side vertical line "H" and "A". Refer to "MSDS458Week3Assighment1.html" for output images of input layer to the hidden layer for each hidden layer node.

Figure2 : Output image of input layer to the hidden layer for each hidden layer node for alphabet "X"



By analyzing the hidden node activations and their relationship to the input and trained weights, it has been possible to visualize how each node segments the output space by what could be thought of as features of the letters. The nodes generate different segmentations that when combined work with the weights going from the hidden layer to the output layer to ultimately identify the correct letter. It will be interesting to see how network with multiple hidden layers will learn to identify features in complex images.

Reference:

Week 3: A.1 First Research/Programming Assignment Instructions.

Week 3: Readings and Resources

Chollet, F. (2018). Deep learning with Python. Shelter Island, NY: Manning Publications.

https://dzone.com/articles/the-very-basic-introduction-to-feed-forward-neural

Code Adopted from simple_5x5-letter-classification_tutorial-code_v7_2017-10-15.py & F. Challot (2018), Deep Learning with Python (Manning)