Basic Neural Network Example

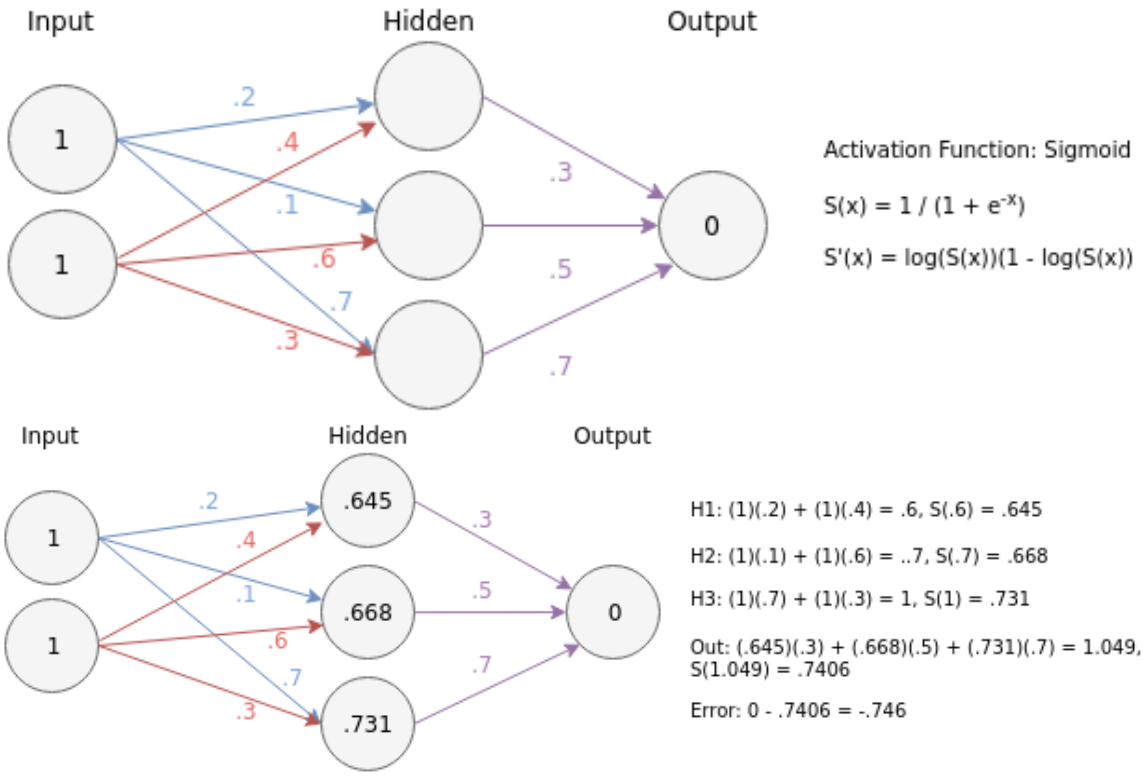
Joseph Blubaugh

08 November 2016

Basic Neural Net

Neural networks are a class of statistical learning models that are well suited for large datasets of continuous variables. They are a series of interconnected nodes and weights that iterate on training data to update weights and minimize training errors. There are several types of neural nets and the type that I have used so far is the feed forward neural net, meaning that data is fed into the model through the input nodes, through the hidden layers, to the output nodes. Other more complex models may have a recursive function that feeds errors back through the model.

The following diagrams are an example of a basic neural net with 2 predictor variables (input), 3 hidden nodes, and 1 output node. Hidden layer weights are calculated using the sum product of the input values and the starting weights which are chosen randomly.



Neural networks require an activation function to transform the input into the appropriate output. In the case of this analysis I am trying to predict texting events so I only want predictions between 0 and 1. An appropriate activation function for this is sigmoid (S(x)).

Weights are initially chosen randomly so neural nets have an iterative process to minimize error by feeding the training error back through the model. The change for each iteration

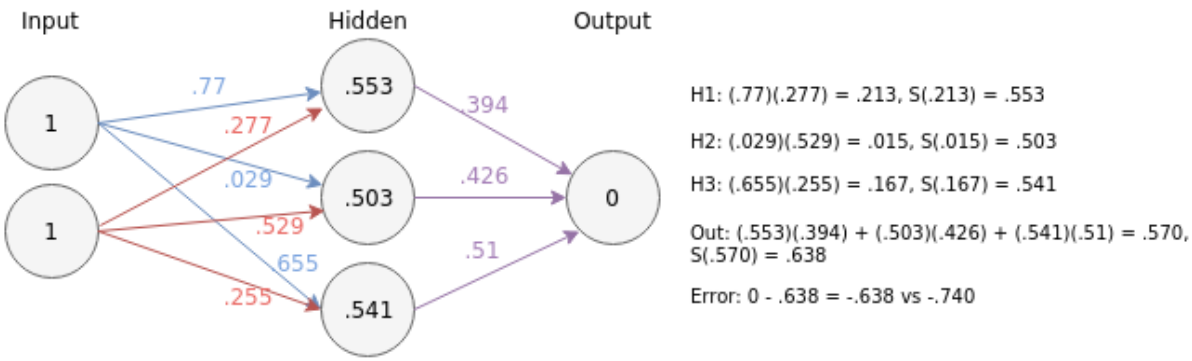


Figure 1: Hidden Weights

is calculated by multiplying the first derivative of the activation function by the most recent training error.

The following steps complete the update for a single training iteration:

```
# Calculate the weight change for the weights between the hidden nodes and output node:
# S'(1.049) = S(1.049)(1 - S(1.049)) * -.7406 = -.1619

# Update the Weights between the Hidden and Output Nodes:
# (-.1619) / [.645, .668, .732] = [-.251, -.242, -.221]

# W7: .645 - .251 = .394
# W8: .668 - .242 = .426
# W9: .731 - .221 = .510

# Calculate the change of the hidden weights
# Divide the weight change by the original [w7, w8, w9] weights times S'(w3, w4, w5)
# -.1619 / [.3, .5, .7] * S'([.6, .7, .1]) = [-.1234, -.0717, -.0454]

# Update the weights between the input nodes and hidden nodes
# [-.1234, -.0717, -.0454] / [1, 1]
```

The diagram below shows the updated neural net for one iteration of the training cycle. It shows that the updated error is slightly smaller than the original training error using random weights.