Neural Networks

# Part 1 Implementation

The code for implementing this homework is contained in hw5.py, hw5data.py, feedforward.py, and neuralnetwork.py

The training, testing, and utility classes such as neural nodes are contained within neuralnetwork.py. The feedforward network itself is implemented in feedforward.py. To make lookup more intuitive and easy, each node contains its own copy of weights coming into the node as well as out. This means that we have two copies of the same weight. These weights are both updated during the weight update process. Since the theta and x’s are assigned accordingly, the weights will undergo the same mathematical operations and stay the same.

Our images are 32 x 30 pixels; thus, our input layer has 960 nodes, our hidden layer has 100 nodes, and finally our output has 1 node.

The homework prompt has indicated that the values of each image pixel is a grayscale float from 0 to 1. When I used python’s opencv library to read the images, I found that the values were integers ranging from 0 to around 32 (it varies). Thus, during the data loading and preprocessing stage, I normalized the image values.

Additionally, the prompt has asked me to create initial weights as random numbers ranging from -1000 to 1000. During implementation, I realized that these weights would create overflow in python’s exp() function, and give me node-x’s that are only either 1 or 0. This is problematic, since during our propagation, theta is calculated as

The term creates a problem when all x’s are 1 or 0. Because evaluating this will yield 0 no matter what. In this case, our nodes would almost never ‘learn’ by updating.

Instead, I chose to initialize weights as a random float from -1 to 0.

I tested the neural network with the test set before training, and once more after training. This gives us a good measure of how much improvement our neural network gained from experience.

Before training, the testing sum-of-square error was around 60.

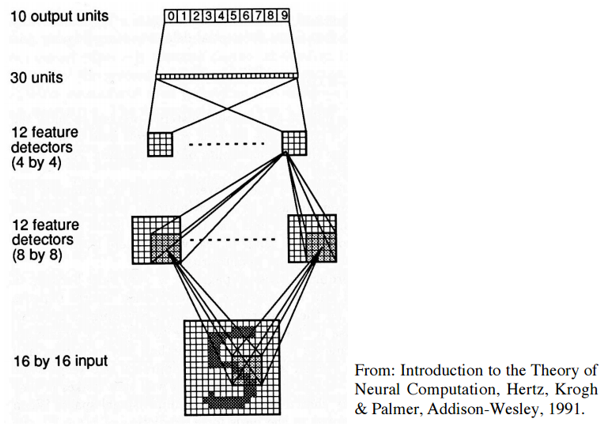
After testing of 20 epochs (training takes a very long time, so I chose 20 for the time being…which runs in several minutes) the error went down to around 6.

These values are not always the same, since initial weights are created at random. However, it’s clear that there were considerable improvements from training.

# Part 2 Software Familiarization

# Application

During my research for scribe notes, I saw various interesting ways neural networks are used to identify and predict time series and patterns. It’s interesting that this homework is very like one of the applications I found for hand-written digit recognition, which has an architecture like the following:



In addition, I saw some interesting examples online of using a feed-forward neural network to learn from a set of words from various languages, and can correctly guess what language any given word came from.