Adam Jaworski

CS 1675

4/7/2014

Assignment 3

**The following experiments were carried out on the bitmap dataset**

Firstly, the data was read in. The first 64 values (0-16) of a line are treated as a 64-element input vector, while the 65th and final value on a line is the digit (0-9) represented by the vector. Every line is a sample.

Once the samples were read in, they were randomly divided into two sets: a training set (75% of the samples) and a test set (25% of the samples). The training set is naturally used for training, while the test set is what we use to evaluate a network’s performance at correctly classifying the samples.

There were three networks being tested. The first was a network with no hidden layers at all; the 64 input units were connected directly to the 10 output units.

The second network had one hidden layer, which contained 10 units, the same size as the output layer. The 64 input units were also conceptually divided into 4 sets of 16, each of which was connected to the hidden layer, which connected to the output layer.

The third and final network featured three hidden layers. The first hidden layer had 32 units, the second had 16, and the third had 10. Also, this time the 64 input units were conceptually divided into 8 sets of 8, each of which was connected to the first hidden layer. The first hidden layer connected to the second, the second connected to the third, and the third hidden layer connected to the output layer. The idea behind breaking up the input into eight sets was that each set would correspond to a row of the bitmap. I thought there may be some benefit to looking at the data by row as opposed to looking at it as a whole.

For learning, a back-propagation trainer was used (a BackpropTrainer object from pybrain). This would train a network on the training data for 20 epochs, which seemed sufficient for convergence. From there, training would continue for 5 more epochs, although this time the results from the test data would be taken as the five trials.

Results for Network with no Hidden Layer:

%Incorrect

Trial 1: 89.42

Trial 2: 89.42

Trial 3: 89.42

Trial 4: 89.42

Trial 5: 89.42

Results for Network with one Hidden Layer:

%Incorrect

Trial 1: 79.37

Trial 2: 78.32

Trial 3: 81.47

Trial 4: 79.27

Trial 5: 78.95

Results for Network with three Hidden Layers:

%Incorrect

Trial 1: 22.62

Trial 2: 23.56

Trial 3: 23.35

Trial 4: 23.04

Trial 5: 22.41

To compare the performance of the no-hidden-layer and one-hidden-layer networks, I performed a paired-t test. Here are their differences in %incorrect for each of the five trials (with the one-hidden-layer result subtracted from the no-hidden-layer result):

Difference

Trial 1: 10.05

Trial 2: 11.1

Trial 3: 7.95

Trial 4: 10.15

Trial 5: 10.47

Average difference (sample mean) = 9.944

Sample variance (s^2) = 1.41068

s = 1.1877

s/sqrt(k) = .53116 (k is trials, = 5)

For 5 trials and 95% confidence, t-value is 2.776

Confidence interval = [sample mean – (t\*(s/sqrt(k))), sample mean + (t\*(s/sqrt(k)))]

Confidence interval = [8.4695, 11.4185]

This result means we can say that, with 95% certainty, the no-hidden-layer network’s % incorrect (more than the one-hidden-layer network) lies somewhere within that range. Because this range does not include 0, we can say that the no-hidden-layer network has inferior performance to the one-hidden-layer network (again, with 95% certainty).

To compare the performance of the three-layer and one-layer networks, I performed another paired-t test. Here are their differences in %incorrect for each of the five trials (with the three-hidden-layer result subtracted from the one-hidden-layer result):

Difference

Trial 1: 56.75

Trial 2: 54.76

Trial 3: 58.12

Trial 4: 56.23

Trial 5: 56.54

Average difference (sample mean) = 56.48

Sample variance (s^2) = 1.44675

s = 1.203

s/sqrt(k) = .5379 (k is trials, = 5)

For 5 trials and 95% confidence, t-value is 2.776

Confidence interval = [sample mean – (t\*(s/sqrt(k))), sample mean + (t\*(s/sqrt(k)))]

Confidence interval = [54.987, 57.973]

This result demonstrates an enormous improvement from my one-layer network to my three-layer one. With 95% confidence, the result of (one-layer % incorrect – three-layer % incorrect) will lie somewhere within that range. Again, the range does not contain zero, and so we can say that my three-layer network offers better performance than my one-layer network.

Since the one-layer network is superior to the no-layer one, it now logically follows that the three-layer network is also superior to the no-layer one.

I tried a few different configurations for the third network. In the same spirit as my idea to look at the bitmap by rows, my first attempt involved looking at the bitmap as pairs of two bits. Then it would look at pairs of pairs, then pairs of pairs of pairs, and so on. This meant I had 32 units in my first hidden layer, 16 in my second, 8 in my third, 4 in my fourth, 2 in my fifth, and 1 in my sixth, which then connected to the output layer. However, when I tested this network I found that it performed no better than the network with no hidden layers at all.

I suspected that the problem may have been that I was squeezing the data into two few units in my later hidden layers. Earlier, when I had been experimenting with the number of hidden layer units for the one-layer network, I discovered that having a small number of units offered no improvement over the no-layer network. When I increased the number of units to 10, I saw improvement.

Noticing the similarity in the situations, I decided I should prevent my network from becoming too bottlenecked in any one layer. The resulting network was sort of a hybrid between my original idea for the third network and what I settled on for the one-layer one.

The results still probably aren’t good enough for this network to be practical in real-world situations, but it achieved results vastly superior to my one-layer network.