Testing the Network Against the Whole Dataset!

Given below is the code to see how the neural network performs against the rest of the dataset.

We keep a score so we can later see if our own ideas for improving the learning worked, and also to compare with how well others have done this. It's easiest to look at the following code and talk through it:

```
# test the neural network

# scorecard for how well the network performs, initially empty
scorecard = []

# go through all the records in the test data set
for record in test_data_list:
    # split the record by the ',' commas
    all_values = record.split(',')
    # correct answer is first value
    correct_label = int(all_values[0])
    print(correct_label, "correct label")
    # scale and shift the inputs
    inputs = (numpy.asfarray(all_values[1:]) / 255.0 * 0.99) + 0.01
    # query the network
    outputs = n.query(inputs)
    # the index of the highest value corresponds to the label
    label = numpy.argmax(outputs)
    print(label, "network's answer")
    # append correct or incorrect to list
    if (label == correct_label):
        # network's answer matches correct answer, add 1 to scorecard
        scorecard.append(1)
    else:
        # network's answer doesn't match correct answer, add 0 to scorecard
        scorecard.append(0)
        pass

    pass
```

Before we jump into the loop which works through all the test data set records, we create an empty list, called *scorecard*, which will be the scorecard that we update after each record.

You can see that inside the loop, we do what we did before, we split the text record by the commas to separate out the values. We keep a note of the first

value as the correct answer. We grab the remaining values and rescale them,

so they're suitable for querying the neural network. We keep the response from the neural network in a variable called *outputs*.

Next is the interesting bit. We know the output node with the largest value is the one the network thinks is the answer. The index of that node, that is, its position, corresponds to the label. That's a long way of saying, the first element corresponds to the label 0, and the fifth element corresponds to the label 4, and so on. Luckily there is a convenient *numpy* function that finds the largest value in an array and tells us its position, `numpy.argmax()`. You can read about it online here. If it returns 0 we know the network thinks the answer is zero, and so on.

That last bit of code compares the label with the known correct label. If they are the same, a 1 is appended to the scorecard. Otherwise, a 0 is appended. I've included some useful `print()` commands in the code so that we can see for ourselves the correct and predicted labels. The following shows the results of this code, and also of printing out the scorecard.

```
# test the neural network

# scorecard for how well the network performs, initially empty
scorecard = []

# go through all the records in the test data set
for record in test_data_list:
    # split the record by the ',' commas
    all_values = record.split(',')
    # correct answer is first value
    correct_label = int(all_values[0])
    print(correct_label, "correct label")
    # scale and shift the inputs
    inputs = (numpy.asfarray(all_values[1:]) / 255.0 * 0.99) + 0.01
    # query the network
    outputs = n.query(inputs)
    # the index of the highest value corresponds to the label
    label = numpy.argmax(outputs)
    print(label, "network's answer")
    # append correct or incorrect to list
    if (label == correct_label):
        # network's answer matches correct answer, add 1 to scorecard
        scorecard.append(1)
    else:
        # network's answer doesn't match correct answer, add 0 to scorecard
        scorecard.append(0)
        pass

    pass
print(scorecard)
```

Not so great this time! We can see that there are quite a few mismatches. The final scorecard shows that out of ten test records, how many were guessed right by our neural network. The code will always generate a different result everytime you run it, try it yourself! Let's finish off with some code to print out that test score as a fraction.

```
# calculate the performance score, the fraction of correct answers
scorecard_array = numpy.asarray(scorecard)
print ("performance = ", scorecard_array.sum() / scorecard_array.size)
```

This is a simple calculation to work out the fraction of correct answers. It's the sum of 1 entries on the scorecard divided by the total number of entries, which is the size of the scorecard. Let's see what this produces in the next lesson!