Plotting the Data Points

Plotting some of the data points to see how digits are formed.

First, we mustn't forget to import the Python extension libraries which will help us with arrays and plotting:

```
import numpy
import matplotlib.pyplot
%matplotlib inline
```

Look at the following 4 lines of code. The variables have been colored to make it easier to understand which data is being used where.

```
all_values = data_list[0].split(',')
image_array = numpy.asfarray(all_values[1:]).reshape((28,28))
matplotlib.pyplot.imshow(image_array, cmap='Greys', interpolation='None')
matplotlib.pyplot.savefig("output/samplePlot2.png")
```

The first line takes the first record from `data_list[0]` that we just printed out, and splits that long string by its commas. You can see the `split()` function doing this, with a parameter telling it which symbol to split by. In this case that symbol is the comma. The results will be placed into `all_values`. You can print this out to check that is indeed a long Python list of values.

The next line looks more complicated because there are several things happening on the same line. Let's work out from the core. The core has that `all_values` list but this time the square brackets `[1:]` are used to take all except the first element of this list. This is how we ignore the first label value and only take the remaining 784 values. The `numpy.asfarray()` is a *numpy* function to convert the text strings into real numbers and to create an array of those numbers. Hang on - what do we mean by converting the text string into numbers? Well, the file was read as text, and each line or record is still text. Splitting each line by the commas still results in bits of text. That text could be

Splitting each line by the commas still results in bits of text. That text could be the word “apple”, “orange123” or “567”. The text string “567” is not the same as a number 567. That’s why we need to convert text strings to numbers, even if the text looks like numbers. The last bit .reshape((28,28)) makes sure the list of number is wrapped around every 28 elements to make a square matrix 28 by 28. The resulting 28 by 28 array is called image_array. Phew! That was a fair bit happening in one line.

The third and fourth line simply plots the image_array using the `imshow()` and outputs the result using `savefig()` function just like we saw earlier. This time we did select a greyscale color palette with `cmap='Greys'` to better show the handwritten characters.

If you look at the output, you can see the plotted image is a 5, which is what the label said it should be. If we instead choose the next record `data_list[1]` which has a label of 0, and look at the output, we get the following image.

```
all_values = data_list[1].split(',')
image_array = numpy.asfarray(all_values[1:]).reshape((28,28))
matplotlib.pyplot.imshow(image_array, cmap='Greys', interpolation='None')
matplotlib.pyplot.savefig("output/samplePlot2.png")
```

You can tell easily the handwritten number is indeed zero.