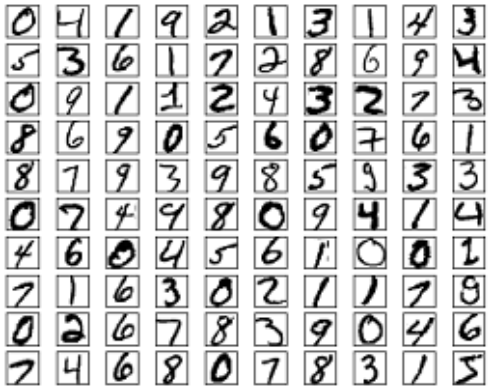
Assignment 2

Lab Assignment 2

"Is that a 1 or a 7?"

Even though the United States Postal Service, as an organization, was formed in 1971, it traces its roots back to the *Post Office Department*, an organization formed in 1792 by President Benjamin Franklin. It later evolved into a cabinet-level department in 1872, before finally being transformed into the USPS we know today in 1971, as an agency of the U.S. government.

Back in the day, all mail was hand read and delivered. Even up the turn of the 20th century, antiquated techniques such as the pigeonhole method from colonial times were used for mail-handling. During the 1950's, the post office started intense research on the coding systems used in many other countries and started down the process of automation. In 1982, the first computer-driven, OCR machine got installed in Los Angeles, and by the end of 1984, over 250 OCRs machines were installed in 118 major mail processing centers across the country and were processing an average of 6,200 pieces of mail per hour.



Nowadays, the Postal Service is one of the world leaders in optical character recognition technology with machines reading nearly +98 percent of all hand-addressed letter mail and +99.5 percent of machine-printed mail, with a single tray sorting machines capable of sorting more than 18 million trays of mail per day.

Let's see if it's possible for you to train a support vector classifier on your computer in a few seconds using machine learning, and if your classification accuracy is similar or better than the advertised USPS stats. For this lab, you'll be making use of the Optical Recognition of Handwritten Digits dataset, provided courtesy of UCI's Machine Learning Repository.

1. Fully review the starter code stored in Module6/**assignment2.py**. If you have any questions about it, please ask them on the forum before you submit your lab answers. The dataset for the lab is stored at /Module6/Datasets/**optdigits.tes** and /Module6/Datasets/**optdigits.tra**. Check out the official dataset page at the UCI ML Repository to figure out why there are two files.
2. Make the requisite changes to get the project running, by providing the path to the .tes and .tra files.
3. Train your SVC classifier with the parameters provided, and keep testing until you're able to beat the classification abilities of the USPS.
4. Remember how important having a lot of samples is for machine learning? Try tossing out 96% of your samples, and see how it affects the accuracy of your highest accuracy support vector classifier.

5. Answer the questions below.

Lab Question

1 point possible (graded)

In this lab, you must complete a series of steps in order to beat the USPS high score for accuracy. What was your accuracy score, as displayed by your assignment, when you first beat the USPS?

Submit You have used 0 of 2 attempts

If you're up for a challenge, check out another handwritten digits datasets, such as The MNIST Database of handwritten digits, and Handwritten Digit Recognition to see how good you can get your classifier to perform on them.

If you need source code to load MNIST - formatted data, such as from the above two links, use the code below:

```
def load(path_img, path_lbl):
  from array import array
  import struct

  with open(path_lbl, 'rb') as file:
    magic, size = struct.unpack(">II", file.read(8))
    if magic != 2049:
      raise ValueError('Magic number mismatch, expected 2049, got {0}'.format(magic))
    labels = array("B", file.read())

  with open(path_img, 'rb') as file:
    magic, size, rows, cols = struct.unpack(">IIII", file.read(16))
    if magic != 2051:
      raise ValueError('Magic number mismatch, expected 2051, got {0}'.format(magic))
    image_data = array("B", file.read())

  images = []
  for i in range(size): images.append([0] * rows * cols)
  for i in range(size): images[i] = np.array(image_data[i * rows * cols:(i + 1) * rows
* cols]).reshape(28,28)[::divisor,::divisor].reshape(-1)
  return pd.DataFrame(images), pd.Series(labels)



X, y = load('digits.data', 'digits.labels')
```

You can set `divisor` to any int, e.g. 1, 2, 3. If you set it to 1, there will be no resampling of the image. If you set it to two or higher, the image will be resamples by that factor of pixels. This, in turn, speeds up training but may reduce overall accuracy.