MNIST digits

May 8, 2019

```
In [172]: import pickle, gzip, numpy as np
          import matplotlib.pyplot as plt
          import matplotlib.cm as cm
          import math
          import random
```

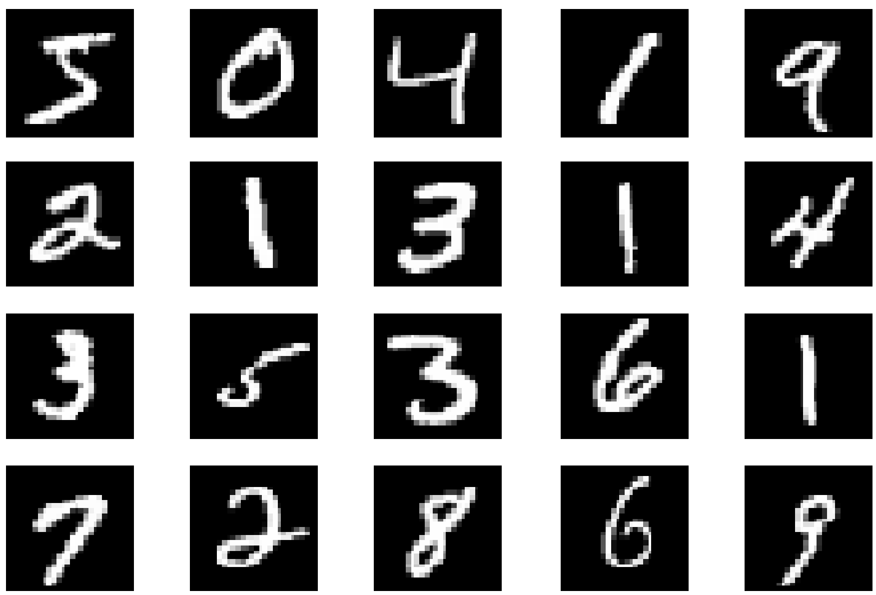
0.0.1 Importing MNIST database:

```
In [173]: def plotImages(X):
              if X.ndim == 1:
                  X = np.array([X])
              numImages = X.shape[0]
              numRows = math.floor(math.sqrt(numImages))
              numCols = math.ceil(numImages/numRows)
              for i in range(numImages):
                  reshapedImage = X[i,:].reshape(28,28)
                  plt.subplot(numRows, numCols, i+1)
                  plt.imshow(reshapedImage, cmap = cm.Greys_r)
                  plt.axis('off')
              plt.show()

          def readPickleData(fileName):
              f = gzip.open(fileName, 'rb')
              data = pickle.load(f, encoding='latin1')
              f.close()
              return data

          def getMNISTData():
              trainSet, validSet, testSet = readPickleData('mnist.pkl.gz')
              trainX, trainY = trainSet
              validX, validY = validSet
              trainX = np.vstack((trainX, validX))
              trainY = np.append(trainY, validY)
              testX, testY = testSet
              return (trainX, trainY, testX, testY)

          def plotImagesHorizontal(X):
```

```
    if X.ndim == 1:
        X = np.array([X])
    numImages = X.shape[0]
    numRows = 1
    numCols = numImages
    for i in range(numImages):
        reshapedImage = X[i,:].reshape(28,28)
        plt.subplot(numRows, numCols, i+1)
        plt.imshow(reshapedImage, cmap = cm.Greys_r)
        plt.axis('off')
    plt.show()

(trainX, trainY, testX, testY) = getMNISTData()
```

We print out the first 20 digits of the MNIST training dataset to get a feel for the dataset.

```
In [174]: plotImages(trainX[0:20,:])
```



For our first proof of concept, we will store one of each of the 9 digits in associative memory.

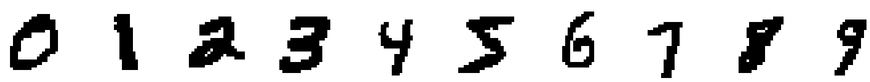
```
In [175]: digits = [1, 6, 5, 7, 26, 0, 18, 42, 17, 43]

          plotImagesHorizontal(np.array([trainX[i] for i in digits]))
```

Pre-process the data so that the digit arrays are composed ot 1s and -1s.

```
In [176]: N = 784 # the mnist digits are 28 by 28 pixels and 28^2 = 784
          s = np.zeros(shape=(784,784))

          # converts digits to 1s and -1s
          return_abs = lambda x: -1 if x > 0 else 1

          # grabs the absolute values for each item in the array for mnist digit
          get_abs = lambda x: np.array([return_abs(i) for i in trainX[x]])
```

Plot the images to see what they look like after processing.

```
In [177]: plotImagesHorizontal(data)
```



0.0.2 Storing Information:

We generate the coupling coefficients according to the following equation:

$$s_{ij} = \frac{1}{N} \sum_{m=1}^{M} \sigma_i^m \sigma_j^m \tag{1}$$

```
In [335]: # stores one of each digit
          data = np.array([get_abs(i) for i in digits])
          # data = trainX[0:20]

          # grabs the coupling coefficients
          for i in range(784):
              for j in range(784):
                  total = 0
                  for k in range(len(data)):
                      total += data[k][i] * data[k][j]
                  s[i][j] = 1/784 * total
```

0.0.3 Retrieving information:

$$y_i = h(x_i) = sgn(x_i) \tag{2}$$

$$u_i = sgn\left(\sum_{j=1}^{N} s_{ij} y_j\right) \longrightarrow \left(\sum_{j=1}^{N} s_{ij} sgn(x_j)\right) \tag{3}$$

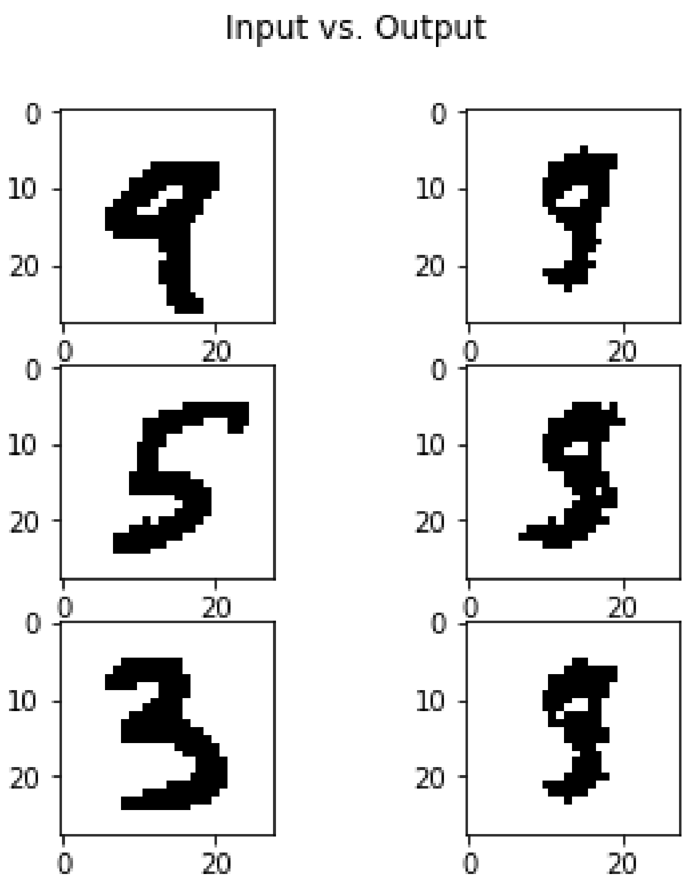
$$\frac{\partial x_i}{\partial t} = -x_i + sgn\left(\sum_{j=1}^{N} s_{ij} sgn(x_j)\right) \tag{4}$$

```
In [231]: def testDigit(input_digit):
              test = np.array([return_abs(i) for i in input_digit])
              V = test # this is the initial state (where the tested image goes)
              new_V = np.ones(784)

              for i in range(784):
                  total_sV = 0
                  for j in range(784):
                      total_sV += s[i][j] * V[j]

                  if total_sV >= 0:
                      new_V[i] = 1
                  else:
                      new_V[i] = -1

              V = new_V
              V_resized = np.array(V).reshape(28, 28)
              test_resized = np.array(test).reshape(28, 28)

              # returns array of input digit and array of retrieved digit
              return test_resized, V_resized
```

0.0.4 Test memory retrieval:

```
In [336]: # testing retrieval of digit 9 from memory
          input_9, output_9 = testDigit(trainX[4])

          # testing retrieval of digit 5 from memory
          input_5, output_5 = testDigit(testX[23])

          # testing retrieval of digit 5 from memory
          input_3, output_3 = testDigit(testX[32])

          f = plt.figure()
          f.set_size_inches(5, 5)
          f.suptitle("Input vs. Output", fontsize=12)

          f.add_subplot(321)
          plt.imshow(input_9, cmap = cm.Greys_r)

          f.add_subplot(322)
```

```
plt.imshow(output_9, cmap = cm.Greys_r)

f.add_subplot(323)
plt.imshow(input_5, cmap = cm.Greys_r)

f.add_subplot(324)
plt.imshow(output_5, cmap = cm.Greys_r)

f.add_subplot(325)
plt.imshow(input_3, cmap = cm.Greys_r)

f.add_subplot(326)
plt.imshow(output_3, cmap = cm.Greys_r)
plt.show()
```



Here we can see how the digits bleed into each other due to interference from the network.

```
In [337]: # stores 20 digits
          data = trainX[0:20]
```

```
# grabs the coupling coefficients
for i in range(784):
    for j in range(784):
        total = 0
        for k in range(len(data)):
            total += data[k][i] * data[k][j]
        s[i][j] = 1/784 * total
```

```
In [333]: for i in [0, 4]:
              inputX, outputX = testDigit(trainX[i])

              f = plt.figure()
              f.set_size_inches(2, 2)
              f.add_subplot(121)
              plt.imshow(inputX, cmap = cm.Greys_r)
              f.add_subplot(122)
              plt.imshow(outputX, cmap = cm.Greys_r)
              plt.show()
```