2019summer-assignment4

July 22, 2019

In [222]:
```
import os
import numpy as np
import matplotlib as plt
import matplotlib.pyplot
import random as rd
from sklearn.model_selection import train_test_split
%cd '/Users/xiaoyingliu/desktop'
```

```
/Users/xiaoyingliu/Desktop
```

In [223]:
```
#1. [20pts] Write an implementation of the perceptron learning algorithm that first
#digit 0 and then for the digit 1. Start with random weights from a normal distribut
#average accuracy on blocks of 25 items and plot this accuracy until you think it wo
DIM = (28,28)
def load_image_files(n, path="images/"):
    # helper file to help load the images
    # returns a list of numpy vectors
    images = []
    for f in os.listdir(os.path.join(path,str(n))):
        # read files in the path
        p = os.path.join(path,str(n),f)
        if os.path.isfile(p):
            i = n.loadtxt(p)
            assert i.shape == DIM # just check the dimensions here
            # i is loaded as a matrix, but we are going to flatten it into a singlev
            images.append(i.flatten())
    return images
```

In [224]:
```
A = load_image_files(0)
B = load_image_files(1)
```

```
---------------------------------------------------------------------------

AttributeError                            Traceback (most recent call last)

<ipython-input-224-50715f08ddcc> in <module>
```

```
----> 1 A = load_image_files(0)
      2 B = load_image_files(1)


<ipython-input-223-5840d039f0d3> in load_image_files(n, path)
     11         p = os.path.join(path,str(n),f)
     12         if os.path.isfile(p):
---> 13             i = n.loadtxt(p)
     14             assert i.shape == DIM # just check the dimensions here
     15             # i is loaded as a matrix, but we are going to flatten it into a single

AttributeError: 'int' object has no attribute 'loadtxt'
```

```
In [225]: #add labels as second column to A and B
          label0=np.zeros((5923,1))
          label1=np.ones((6742,1))
          A_withlabel=np.column_stack((A, label0))
          B_withlabel=np.column_stack((B,label1))

In [75]: N = len(A[0]) # the total sizeassert should be 784

         N == DIM[0]*DIM[1] # just check our sizes to be sure
         # set up some random initial weights
         weights = np.random.normal(0,1,size=N)

In [76]: #show initial weights #784 entries
         np.shape(weights)

Out[76]: (784,)

In [8]: #classification function
        def predict(data_point, weights):
                b = np.dot(data_point, weights)
                a = b>0
                return a*1

In [141]: def update(weights, data_point, labels, alpha=.1):
                predicted = predict(data_point, weights)
                weight_temp = np.zeros(np.shape(weights))
                weight_temp = alpha*(labels-predicted)*data_point
                return weight_temp+weights

In [173]: #in total 12665 images
          AB=np.vstack((A_withlabel,B_withlabel))
          np.shape(AB)

In [211]: #need to randomly shuffle AB,this is a substitution of sampling from AB
          #if label match, corretc+1 total+1 if not total+1 when total =25, acc=correct/total
```

```
        def train_perceptron(data, weights, alpha = .001, iterations = 10):
            for j in range(0, iterations):
                for i in range(0, 12665):
                    weights = update(weights, data[i][:-1], data[i][783:784], alpha)
            return weights

        final_weights=train_perceptron(AB,weights,alpha=0.001,iterations=1)
```

In [213]:
```
        #test accuracy
        def test_all(data, labels, weights):
                a, b = np.shape(labels)
                predicted = predict(data, weights)
                correct = predicted == labels[:,0]

                accuracy = np.sum(correct)/float(a)
                return accuracy
```

In [214]:
```
test_all(AB[:, :-1],AB[:, 783:784] ,final_weights)
```

Out[214]:
```
0.9975523095144098
```

In [ ]:
```
#Conclusion:as the training number of images and iterations increases, accuracy has be
#10 interations, accuracy has been increased to 99.76%
```

In [ ]:
```
#2. [5pts] Does your solution in Q1 converge on 100% accuracy or not? What does this m
#the linear separability of 0 and 1 on this feature space?
#Ans: yes, it means that 0 and 1 in the feature space is linearly seperable, that our
```

In [ ]:
```
#3. [15pts] Reshape (numpy.reshape) your weight vector after training so that it is a
#corresponds to the weight assigned to each pixel in the image. Show a picture of this
#interpret it in a sentence or two. What do large negative and large positive values m
#What do numbers near zero mean? Why does this matrix look the way that it does, in te
#large positive and negative terms are located?
#Ans:
#use heat matrix to display weights. Large values are useful, more predictable. small
#Thus, the location of large numbers and small numbers are an intuitive way of showing
#the specific hand written digit.
```

In [20]:
```
#4. [10pts] What should you expect to happen if you set the elements of the weight ve
#close to zero to be actually zero? Do this for the 10, 20, 30,  780 weight values cl
#absolute value) and plot the resulting accuracies on 1000 random classifications of
#does this tell you about the proportion of the image which is diagnostic about 0 vs
```

```
---------------------------------------------------------------------------

TypeError                                 Traceback (most recent call last)

<ipython-input-20-19b1f85f8ff3> in <module>
```

```
      1
----> 2 fig = plt.figure()
      3 ax1 = fig.add_subplot(121)
      4 ax1.imshow(a, interpolation='bilinear', cmap=cm.Greys_r)
      5 ax2 = fig.add_subplot(122)


    TypeError: 'module' object is not callable
```

In [ ]:
```
#Ans: If we set near zeros to zeros. We would eliminate these location-wise numbers as
#linear seperator. My expectation is that the accuracy might be decreased.
```

In [ ]:
```
#5. [20pts] Next show a matrix of the classification accuracy of each pair of digits a
#Make this a plot (with colors for accuracy rather than numbers). Does it match your i
#which pairs should be easy vs. hard? Why or why not?
#Ans: we could construct an algoritm which can identify all numbers at the same time.
#classifying all numbers, the other is for whenever the classfier gives 2 conflicting
#most reliable prediction result
```

In [74]:
```
#return a set of weights which could classfiy all digits
def all_numbers(data,labels):
        c,d = np.shape(data)
        w = create_weights(data)
        weights = []
        for i in range(0, 10):
                z = same_number(labels, i)
                a = train_perceptron(data, z, w, .001, 4)
                weights.append(a[:,0])
        return np.asarray(weights)
```

In [75]:
```
def one_all(data, weights):
        a = np.dot(data,np.transpose(weights))
        b = len(np.shape(data))
        if b == 1:
                return np.argmax(a)
        return np.argmax(a, axis=1)
```

Out[75]:
```
array([6, 2, 7, ..., 0, 3, 6])
```

In [ ]:
```
#more iterations and moderate learning rate will give out best results, we will do 200
```