COGSCI131–Spring 2019 — Homework 6Solutions

Yuchen Zhou, SID 3034489901

1.

(a) Take one block (25 items) to train and one block to test each iteration.

(b) Code as follows:

```python
import os
import numpy
import random
import pickle
import matplotlib.pyplot as plt

DIM = (28, 28)

def H(num):
    if(num<0):
        return 0
    else:
        return 1

def load_image_files(n, path="images/"):
    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 = numpy.loadtxt(p)
            assert i.shape == DIM
            images.append(i.flatten())
    return images

A = load_image_files(0)
B = load_image_files(1)

N = len(A[0])  # the total size
assert N == DIM[0] * DIM[1]  # just check our sizes to be sure

weights = numpy.random.normal(0, 1, size=N)
AA=A
BB=B
acc=[0]
r=0
fig = plt.figure()
ax = fig.add_subplot(111)
r+=1

print(len(A),len(B))
for i in range(5):
    for i in range(25):
        if (len(AA) != 0 and len(BB) != 0):
            mark = random.randint(0, 1)
        elif (len(AA) == 0 and len(BB) == 0):
            print("step", r, ", empty.")
            break
        elif (len(AA) == 0):
            mark = 1
        else:
            mark = 0
        if (mark):
            k = random.randint(0, len(BB) - 1)
            temp = BB[k]
            BB.pop(k)
        else:
            k = random.randint(0, len(AA) - 1)
            temp = AA[k]
            AA.pop(k)
        if (mark):
            if (not H(numpy.dot(weights, temp))):
                weights += temp
        else:
            if (H(numpy.dot(weights, temp))):
                weights -= temp
    accsum = 0
    for i in range(25):
        if (len(AA) != 0 and len(BB) != 0):
            mark = random.randint(0, 1)
        elif (len(AA) == 0 and len(BB) == 0):
            print("step", r, ", empty.")
            break
        elif (len(AA) == 0):
            mark = 1
        else:
            mark = 0
        if (mark):
            k = random.randint(0, len(BB) - 1)
            temp = BB[k]
            BB.pop(k)
        else:
            k = random.randint(0, len(AA) - 1)
            temp = AA[k]
            AA.pop(k)
        if (mark):
            if (H(numpy.dot(weights, temp))):
                accsum += 1
        else:
            if (not H(numpy.dot(weights, temp))):
                accsum += 1
    acc.append(accsum / 25)
    print(acc[r])
    r+=1
filename = 'weights.data'
f = open(filename, 'wb')
pickle.dump(weights, f)
f.close()
#plt.imshow(numpy.reshape(weights, (28, 28)), cmap=plt.cm.gray)
#plt.show()

ax.plot(acc)
```

```
plt.show()
```

(c) As figure shows, after two iterations accuracy is very close to 1, so there is no need to keep training.

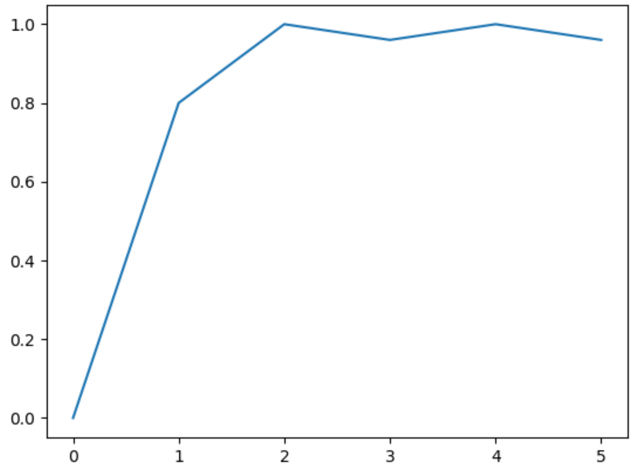

Figure 1: accuracy to number of blocks

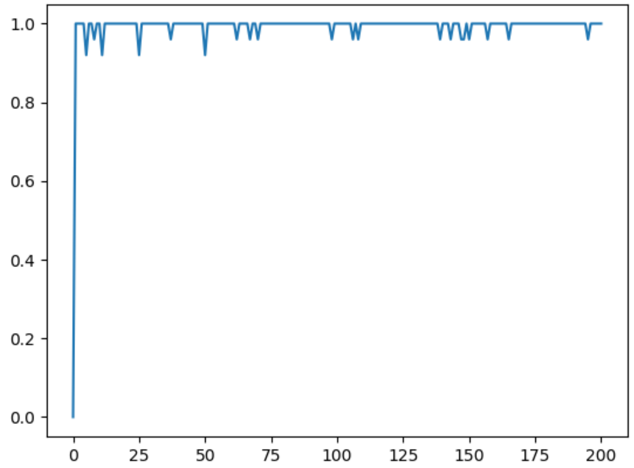

Figure 2: accuracy to number of blocks

(d) accuracy can reach 100%.

2.

Yes, the accuracy is very close to 100%.
It means that we can use a $28 \times 28 - 1 = 783$ dimensions plane to perfectly distinguish 0 from 1 without fault.

3.

(a) Weights image as follows:

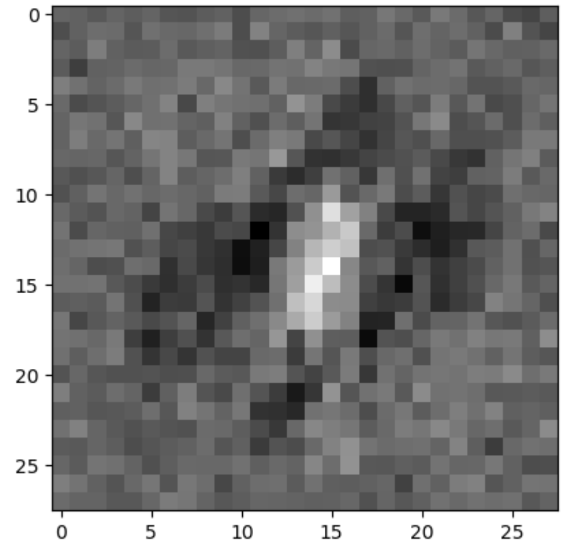

Figure 3: weights in image

(b) This weights vector is close to ”1” and far from ”0”. So in weights image, pixels where ”1” should occur is white and pixels where ”0” should occur is black.

(c) Large negative number pixel means that a typical ”0” is black at that pixel. Large positive number pixel means that a typical ”1” is black at that pixel. Number near 0 means pixel there doesn’t have a significant impact on distinguishing ”0” from ”1”.

(d) The large positive number (white in image) are located in ”1” while large negative number (black in image) are located in ”0”, so it looks in that way.

4.

(a) I expect the accuracy won’t change much if I set the elements of the weight vector which are close to zero to be actually zero.

(b) Use trained weights from question 1. Code:

```
import os
import numpy
import random
import pickle
import matplotlib.pyplot as plt
import heapq

DIM = (28, 28)

def H(num):
    if(num<0):
        return 0
    else:
        return 1

def load_image_files(n, path="images/"):
    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 = numpy.loadtxt(p)
            assert i.shape == DIM
            images.append(i.flatten())
    return images

A = load_image_files(0)
B = load_image_files(1)

acc=[]
r=0
fig = plt.figure()
ax = fig.add_subplot(111)
filename = ’weights.data’
f = open(filename, ’rb’)
weights = pickle.load(f)
marked=numpy.zeros(28*28)
for k in range(10,790,10):
    w2=abs(weights)
    minarr=heapq.nlargest(k, w2)
    for i in range(k):
        for j in range(28*28):
            if(w2[j]==minarr[i]):
                w2[j]=0
    accsum = 0
    AA = A.copy()
    BB = B.copy()
    for p in range(1000):
        if (len(AA) != 0 and len(BB) != 0):
            mark = random.randint(0, 1)
        elif (len(AA) == 0 and len(BB) == 0):
            print("step", r, ", empty.")
            break
        elif (len(AA) == 0):
            mark = 1
        else:
            mark = 0
        if (mark):
            k = random.randint(0, len(BB) - 1)
            temp = BB[k]
            BB.pop(k)
        else:
            k = random.randint(0, len(AA) - 1)
            temp = AA[k]
            AA.pop(k)
        if (mark):
            if (H(numpy.dot(weights, temp))):
                accsum += 1
        else:
            if (not H(numpy.dot(weights, temp))):
                accsum += 1
    acc.append(accsum / 1000)
    print(acc[r])
    r += 1
ax.plot(acc)
plt.show()
#plt.imshow(numpy.reshape(w2, (28, 28)), cmap=plt.cm.gray)
#plt.show()
```

(c) Figure as follows:

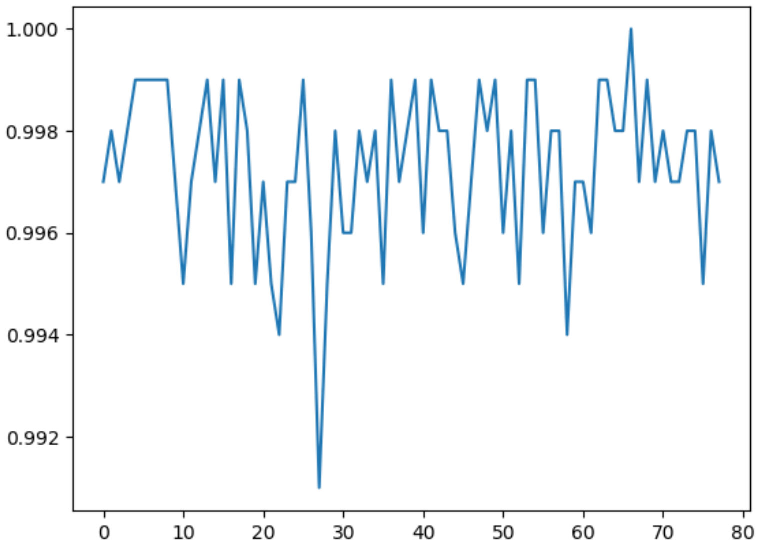

Figure 4: accuracy when number of values close to 0 to be 0

(d) From the plot we can find that changing values close to 0 to 0 can hardly decrease the accuracy. So only tens of points are important when judging whether the number is a ”0” or a ”1”. We can decrease the judging dimensions now.

5.

(a) Train every pair by 20 blocks. Matrix as follows:

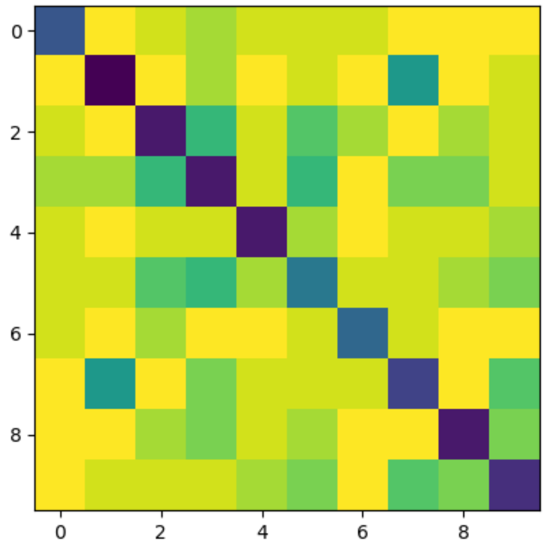

Figure 5: Matrix of accuracy for each pair

(b) From the matrix we notice that it's hard to distinguish one from itself. Also, 1 and 7, 2 and 3, 2 and 5, 3 and 5 are confusion pairs, while 0 and 1, 0 and 7, 0 and 8, 0 and 9, 1 and 2, 1 and 4, . . . , are distinguishable.

(c) 0 and 8 both have circle, 4 and 9 are both formed by a circle and a line. They have similar features so they are hard to distinguish.

(d) Code:

```
import os
import numpy
import random
import pickle
import matplotlib.pyplot as plt
DIM = (28, 28)

def H(num):
    if(num<0):
        return 0
    else:
        return 1

def load_image_files(n, path="images/"):
    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 = numpy.loadtxt(p)
            assert i.shape == DIM
            images.append(i.flatten())
    return images

comp=numpy.zeros((10,10))
for num1 in range(10):
    for num2 in range(num1,10):
        A = load_image_files(num1)
        B = load_image_files(num2)
        N = len(A[0])  # the total size
        assert N == DIM[0] * DIM[1]  # just check our sizes to be sure
        weights = numpy.random.normal(0, 1, size=N)
        AA = A.copy()
        BB = B.copy()
        acc = 0
        for w in range(20):
            for i in range(25):
                if (len(AA) != 0 and len(BB) != 0):
                    mark = random.randint(0, 1)
                elif (len(AA) == 0 and len(BB) == 0):
                    print("empty.")
                    break
                elif (len(AA) == 0):
                    mark = 1
                else:
                    mark = 0
                if (mark):
                    k = random.randint(0, len(BB) - 1)
                    temp = BB[k]
                    BB.pop(k)
                else:
                    k = random.randint(0, len(AA) - 1)
                    temp = AA[k]
                    AA.pop(k)
                if (mark):
                    if (not H(numpy.dot(weights, temp))):
                        weights += temp
                else:
                    if (H(numpy.dot(weights, temp))):
                        weights -= temp
            accsum = 0
            for i in range(25):
                if (len(AA) != 0 and len(BB) != 0):
                    mark = random.randint(0, 1)
                elif (len(AA) == 0 and len(BB) == 0):
                    print("empty.")
                    break
                elif (len(AA) == 0):
                    mark = 1
                else:
                    mark = 0
                if (mark):
                    k = random.randint(0, len(BB) - 1)
                    temp = BB[k]
                    BB.pop(k)
                else:
                    k = random.randint(0, len(AA) - 1)
                    temp = AA[k]
                    AA.pop(k)
                if (mark):
                    if (H(numpy.dot(weights, temp))):
                        accsum += 1
                else:
                    if (not H(numpy.dot(weights, temp))):
                        accsum += 1
            acc=accsum/25
            print(acc)
        comp[num1][num2]=acc
        comp[num2][num1]=acc
fig = plt.figure()
ax = fig.add_subplot(111)
plt.imshow(comp)
plt.show()
```