38_P2_PeerAssessment_GionRubitschung

December 14, 2023

a. Make sure matplotlib is installed on your machine and, in Python, run

```
>>> import numpy as np
>>> import matplotlib.pyplot as plt
```

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
```

b. From Moodle, download the files 200_0s.csv through 200_9s.csv, containing a subset of the public MNIST dataset corresponding to numbers 0 through 9 respectively. In NumPy, load the data corresponding to numbers $1, 2, 3$ using

```
>>> dir = 'INSERT_NAME_OF_DIRECTORY_CONTAINING_FILES'
>>> loadxs = lambda i : np.genfromtxt(
                dir + '/200_' + str(i) + 's.csv',delimiter=",")
>>> xs1 = loadxs(1)
>>> xs2 = loadxs(2)
>>> xs3 = loadxs(3)
```

As a check, run `xs2.shape` which should return `(784,200)`; make sure you understand the role of the numbers 784 ($= 28^2$) and 200. Also run

```
>>> plt.imshow(xs2[:,0].reshape((28,28)),cmap='gray')
```

which should give you the image on the next slide.

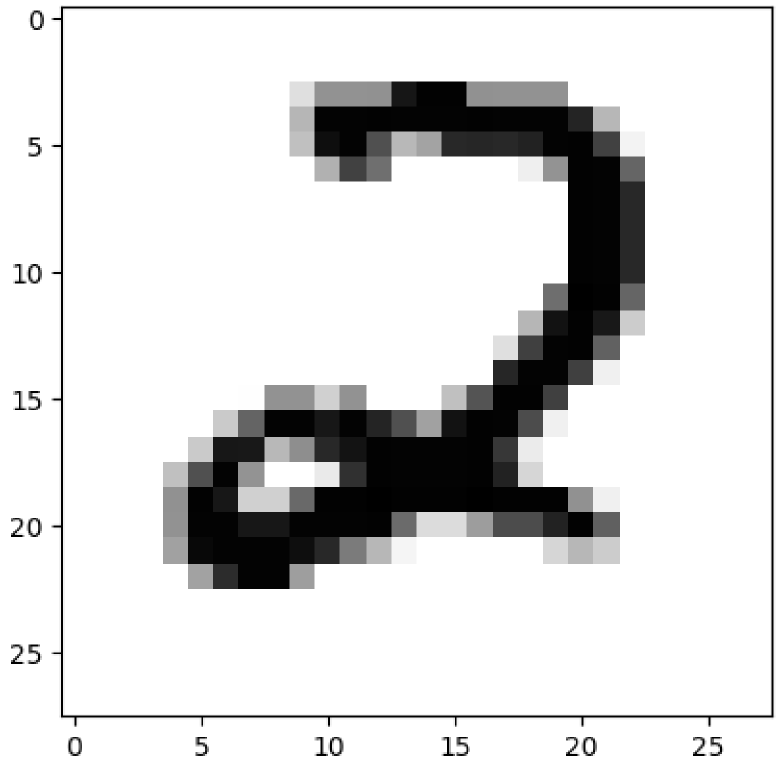
[2]:
```
loadxs = lambda i: np.genfromtxt(f"./mnist_data/200_{i}s.csv", delimiter=",")

xs1 = loadxs(1)
xs2 = loadxs(2)
xs3 = loadxs(3)
```

[3]:
```
xs2.shape
```

[3]:
```
(784, 200)
```

[4]:
```
plt.imshow(xs2[:,0].reshape((28,28)),cmap='gray')
```

[4]: <matplotlib.image.AxesImage at 0x127b18350>



c. Define a 784×600 matrix `xs` corresponding to the block matrix build from the three matrices `xs1`, `xs2`, `xs3`. **Hint.** Use `np.block`.

[5]:
```
xs = np.block([xs1, xs2, xs3])
xs.shape
```

[5]: (784, 600)

d. Compute the mean of each row of `xs` using

```
>>> xmean = np.mean(xs,axis=1)
```

Use `xmean.shape` to check that this really is a vector with 784 entries!

[6]:
```python
xmean = np.mean(xs, axis=1)
xmean.shape
```

[6]: (784,)

e. Subtract the mean, and get zero-mean data, using

```python
>>> Xs = xs - np.outer(xmean, np.ones(600))
```

[7]:
```python
Xs = xs - np.outer(xmean, np.ones(600))
Xs.shape
```

[7]: (784, 600)

f. Now compute the data covariance matrix, and store it as `cov`.
Hint. The product `Xs @ Xs.T` may be useful.

[8]:
```python
n = xs.shape[1]
cov = (Xs @ Xs.T) / (n - 1)
cov.shape
```

[8]: (784, 784)

g. Compute a spectral decomposition of the data covariance matrix using

```python
>>> lambdas, V = np.linalg.eigh(...)
```

[9]:
```python
lambdas, V = np.linalg.eigh(cov)
D = np.diag(lambdas)
```

[10]:
```python
V.shape
```

[10]: (784, 784)

[11]:
```python
D.shape
```

[11]: (784, 784)

h. Determine the two largest eigenvalues.

Since the eigenvalues in D have to be in increasing order, we can just extract the last two values

[12]:
```python
eval_1 = D[783][-1]
eval_2 = D[782][-2]
```

[13]:
```python
eval_1
```

[13]: 6.625891531753323

[14]:
```python
eval_2
```

[14]: 5.116375890772888

i. Extract the columns of V corresponding to the largest and second largest eigenvalue and store them as `vx` and `vy` respectively. Plot them using

```python
>>> plt.imshow(vx.reshape((28,28)),cmap='gray')
>>> plt.imshow(vy.reshape((28,28)),cmap='gray')
```

Note. These are the principal directions.

[15]:
```python
vx = V[:,783]
vy = V[:,782]
```

[16]:
```python
vx.shape
```

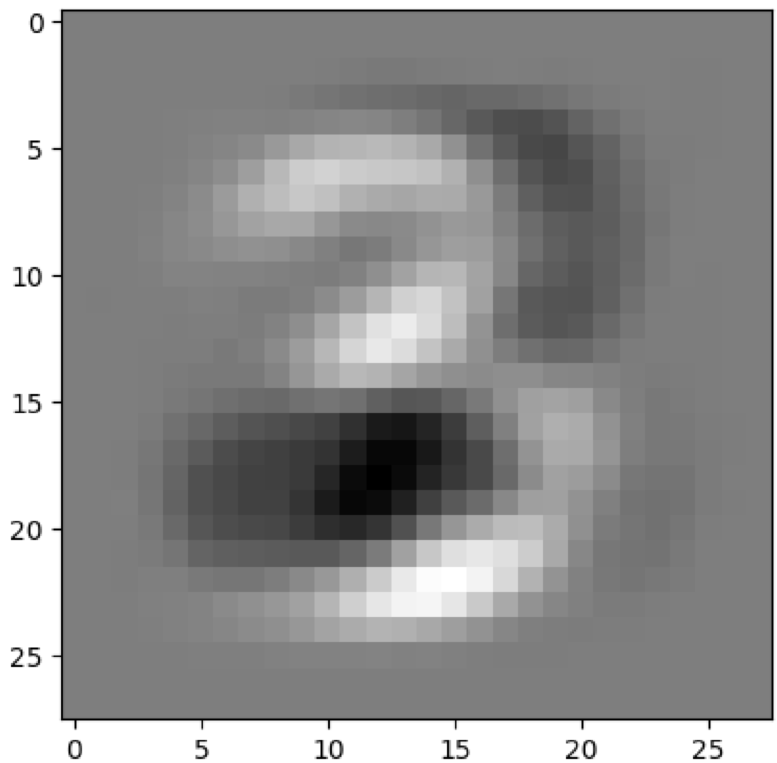
[16]: (784,)

[17]:
```python
vy.shape
```

[17]: (784,)

[18]:
```python
plt.imshow(vx.reshape((28,28)), cmap='gray')
plt.imshow(vy.reshape((28,28)), cmap='gray')
```

[18]: <matplotlib.image.AxesImage at 0x126cebf90>



j. Compute the dot product of each column of `Xs` with `vx` and put the result in a vector `px` of length 600. Similarly, compute the dot product of each column of `Xs` with `vy` and put the result in a vector `py`.
Note. These are the principal components.

[19]:
```
px = np.dot(vx, Xs)
py = np.dot(vy, Xs)
```

[20]:
```
len(px)
```

[20]: 600

[21]:
```
len(py)
```

[21]: 600

k. Make a scatter plot using

```
>>> col = np.block([0*np.ones(200),1*np.ones(200),2*np.ones(200)])
>>> plt.scatter(px,py,c=col)
```

[22]:
```
col = np.block([0 * np.ones(200), 1 * np.ones(200), 2 * np.ones(200)])
plt.scatter(px, py, c=col)
```

[22]: <matplotlib.collections.PathCollection at 0x126d6be90>