Final Project - MAT4373

Tanner Giddings

2024-04-03

```
#!pip install matplotlib
#!pip install pandas
#!pip install tqdm
```

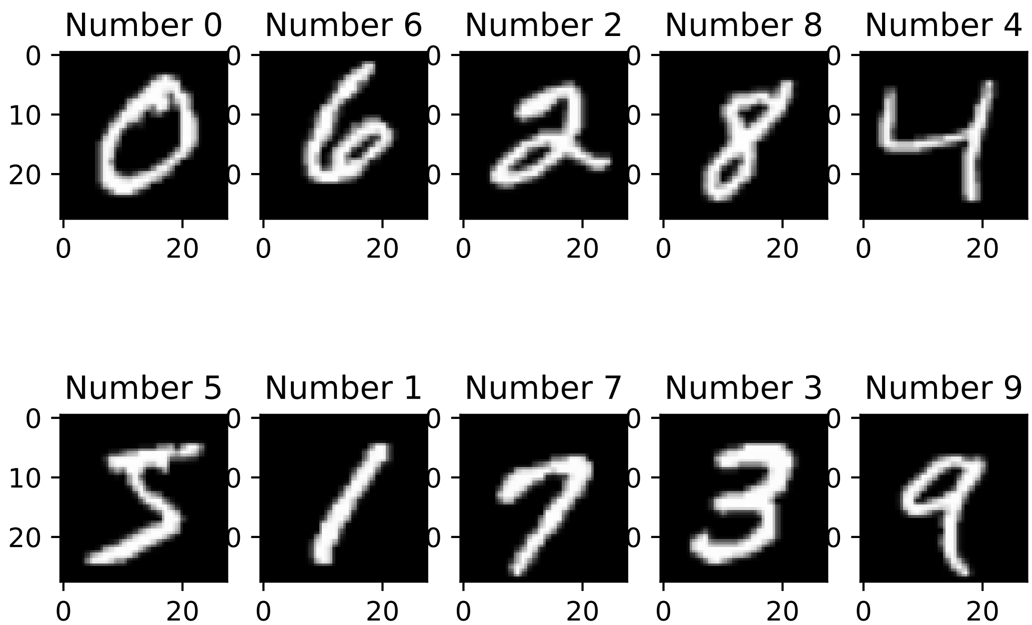
Question 1

```
import matplotlib.pyplot as plt
from scipy.io import loadmat
import pandas as pd

data = loadmat('mnist_all.mat')
data.keys()
```

```
## dict_keys(['__header__', '__version__', '__globals__', 'train0', 'test0', 'train1', 'test1', 'train2
```

```
fig, ax = plt.subplots(2, 5)
for i in range(10):
  ax[i % 2][i % 5].imshow(data[f"train{i}"][0].reshape((28,28)), cmap='gray')
  ax[i % 2][i % 5].set_title(f"Number {i}")
plt.show()
```




```
#Cleaning the data
#Taking too long to run
"""
import pandas as pd

def clean(D):
  df = []
  for i in range(0,10):
    dict = {"pixel" + str(j) : [D[f"train{i}"][k][j] for k in range(len(D[f"train{i}"]))] for j in rang
    dict['Y'] = [i] * len(dict['pixel0'])
    df.append(pd.DataFrame(dict))
  return pd.concat(df)

data_clean = clean(data)
data_clean
"""
```

```
## '\nimport pandas as pd\n\ndef clean(D):\n  df = []\n  for i in range(0,10):\n    dict = {"pixel" + s
```

```
from tqdm import tqdm
def clean(D):
  d = {}
  for i in range(0, 10):
    d["train" + str(i)] = [elem / 255 for elem in D["train" + str(i)] for i in range(0,10)]
  return d
data_clean = clean(data)
```

Question 2

```
import numpy as np

def softmax(x):
  e_x = np.exp(x)
  return e_x / e_x.sum()

def ReLU(x):
  return [max(0, elem) for elem in x]

def forward(X, w, b):
  return np.matmul(X, w) + b

def predict(X, w, b):
  return softmax(ReLU(forward(X, w, b)))

w1 = np.random.rand(28*28, 9)
b1 = np.random.rand(9)

print(predict(np.reshape(data_clean['train0'][0], (28*28)), w1, b1), "\nThis is the predicted outputs fo
```

```
## [9.10412283e-03 1.61684365e-02 7.62623195e-01 3.88820283e-02
##  1.47980334e-01 2.35521130e-02 4.09372231e-04 3.83959224e-04
##  8.96438968e-04]
## This is the predicted outputs for an untrained model
```

```
#np.reshape(data['train0'][0], (28,28))
```

Part 3

Let $L(\overrightarrow{y}, \overrightarrow{p}) = -\sum_{i=0}^{9}(y_i log(p_i) + (1-y_i) log(1-p_i))$ be the loss function at the end of the network for one sample, the variable i here represents each class, and y_i represents the one-hot encoded value of the class, e.g. $y_i = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$ represents 2.

$$
\begin{aligned}
\frac{\delta}{\delta p_i} L(\overrightarrow{y}, \overrightarrow{p}) &= \frac{\delta}{\delta p_i}(-\sum_{i=0}^{9}(y_i log(p_i) + (1 - y_i) log(1 - p_i))) \\
&= -\sum_{i=0}^{9}(\frac{y_i}{p_i} - \frac{1 - y_i}{1 - p_i}) \\
&= -\sum_{i=0}^{9}(\frac{y_i - y_i p_i - p_i + y_i p_i}{p_i(1 - p_i)}) \\
&= -\sum_{i=0}^{9} \frac{y_i - p_i}{p_i(1 - p_i)} \\
&\text{Suppose the class of the sample is } k\text{, so } y_k = 1 \text{ and } y_i = 0 \ \forall i \neq k \\
&= \sum_{i=0, i \neq k}^{9} \frac{1}{1 - p_i} - \frac{1}{p_k}
\end{aligned}
$$