Assignment5_ErenAkgunduz-1

February 18, 2024

1 Assignment 5

1.1 Eren Akgunduz

1.1.1 Deep Learning — 18 February 2024

1.1.2 Link to notebook

[1]:
```
from keras.datasets import mnist
import matplotlib.pyplot as plt
import numpy as np
```

1.2 Problem 1

[2]:
```
(x_train, y_train), (x_test, y_test) = mnist.load_data()
```

[3]:
```
# Number of images in each training and testing set
print(x_train.shape[0], y_train.shape[0])
print(x_test.shape[0], y_test.shape[0])
```

```
60000 60000
10000 10000
```

[4]:
```
# Image width and height
img_ws_hs = {x_train.shape[1], x_train.shape[2], x_test.shape[1], x_test.
 ↪shape[2]}  # it's the same for all so only one value in set
img_pxs = list(img_ws_hs)[0]
print(img_ws_hs, img_pxs)
print(f"{img_pxs} x {img_pxs}")
```

```
{28} 28
28 x 28
```

[5]:
```
# From provided code examples
def img_plt(images, labels):
    "Plots a figure with 10 subplots for each 0-9 digit"
    plt.figure()  # figsize=(15,8)
    for i in range(1,11):
        plt.subplot(2,5,i)
        plt.imshow(images[i-1,:,:],cmap='gray')
```

```
        plt.title('Label: ' + str(labels[i-1]))
    plt.show()
```

[6]:
```
# Shuffle the training set indices
num_train = x_train.shape[0]
train_ind = np.arange(0, num_train)
train_ind_shuffled = np.random.permutation(train_ind)
```

[7]:
```
# Just to confirm
train_ind[:10]
```

[7]:
```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

[8]:
```
train_ind_shuffled[:10]
```

[8]:
```
array([50966,  3978, 48962, 57185, 12814, 32445, 14080, 21062,  1251,
       48477])
```

[9]:
```
x_train = x_train[train_ind_shuffled,:,:]
y_train = y_train[train_ind_shuffled]

# Select 20% of training data as validation set
x_valid = x_train[0:int(0.2 * num_train), :, :]
y_valid = y_train[0:int(0.2 * num_train)]

# The rest of the training set
x_train = x_train[int(0.2 * num_train):, :, :]
y_train = y_train[int(0.2 * num_train):]
```

[10]:
```
try:
    list({x_train.shape[0], y_train.shape[0]})[1]
except IndexError:
    print("Number of images in training set:", list({x_train.shape[0], y_train.
 ↪shape[0]})[0])
    print("Number of images in validation set:", list({x_valid.shape[0],␣
 ↪y_valid.shape[0]})[0])
    print("Number of images in testing set:", list({x_test.shape[0], y_test.
 ↪shape[0]})[0])
else:
    print("Oh no, seems like your data is not split properly ;) try again")
```

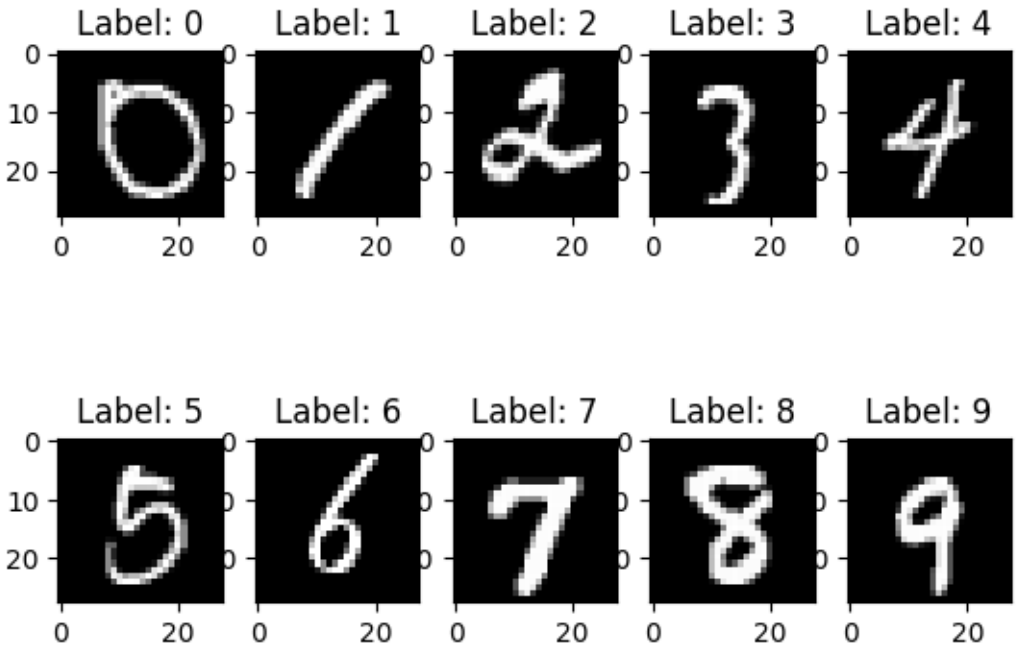
```
Number of images in training set: 48000
Number of images in validation set: 12000
Number of images in testing set: 10000
```

[11]:
```
def plot_all(x_set, y_set, set_name):
    print(f"Selecting 10 images from {set_name} set")
    x_rnd = np.zeros((10, x_set.shape[1], x_set.shape[2]))
```

```
    y_rnd = np.arange(10)
    for digit in range(10):
        x_d = x_set[y_set == digit,:,:]
        x_rnd[digit,:,:] = x_d[0, :, :]  # selecting first digit from the set
    img_plt(x_rnd, y_rnd)
```

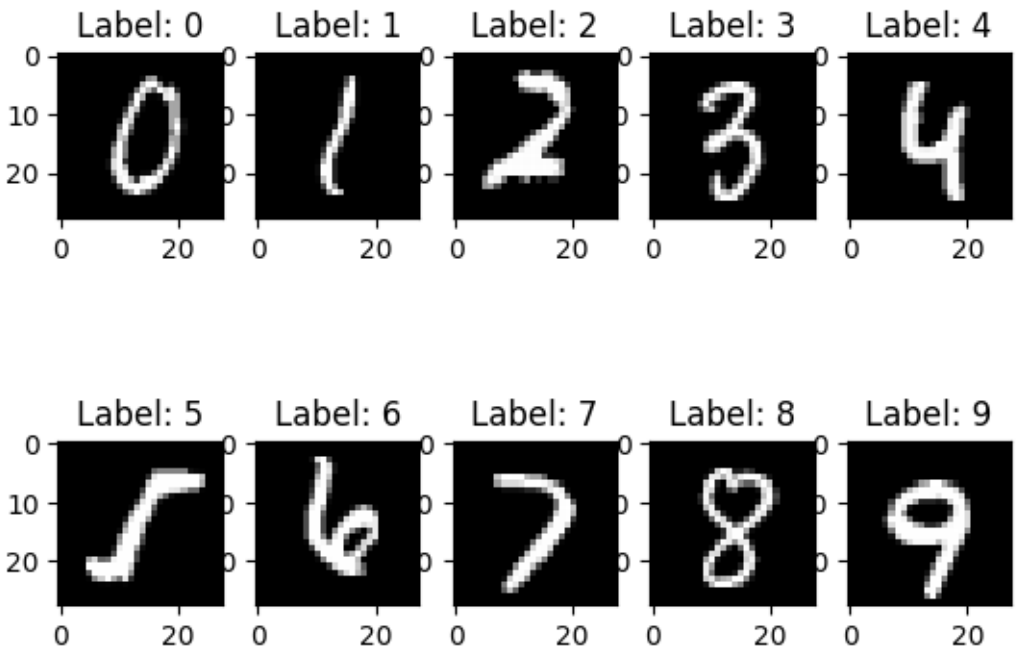
[12]:
```
plot_all(x_train, y_train, "training")
```

```
Selecting 10 images from training set
```



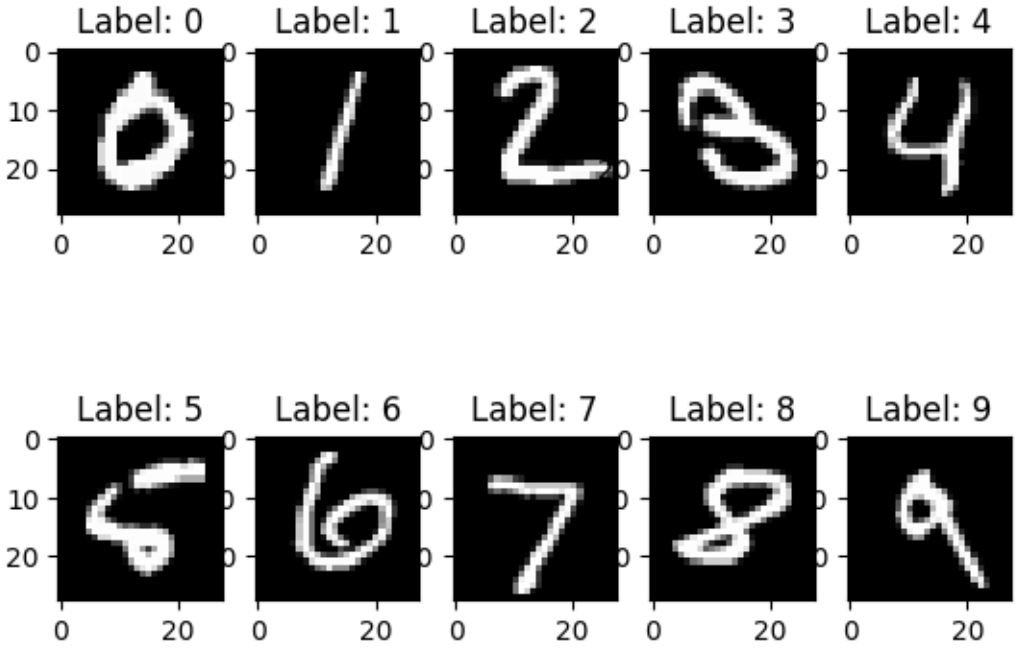
[13]:
```
plot_all(x_valid, y_valid, "validation")
```

```
Selecting 10 images from validation set
```



[14]:
```
plot_all(x_test, y_test, "testing")
```

```
Selecting 10 images from testing set
```



1.3 Problem 2

[15]:
```
def feat_plt(feature, labels, digits):
    plt.figure()
    sample_nums = np.arange(feature.shape[0])
    plt.plot(sample_nums[labels==digits[0]], feature[labels==digits[0]], 'gs',
             sample_nums[labels==digits[1]], feature[labels==digits[1]], 'r^')
    plt.legend([f'Digit {digits[0]}', f'Digit {digits[1]}'])
    plt.xlabel('Sample #')
    plt.ylabel('Average of the 3x3 center grid')
    plt.title('Extrated feature from validation data')
    plt.show()
```

[16]:
```
def pred_fun(features, threshold, digits):
    y_pred = np.ones(features.shape) * digits[0]
    y_pred[features > threshold] = digits[1]
    return y_pred
```

[17]:
```
def acc_fun(labels_actual, labels_pred):
    acc = np.sum(labels_actual==labels_pred) / len(labels_actual) * 100
    return acc
```

[18]:
```
(x_train, y_train), (x_test, y_test) = mnist.load_data()
```

[19]:
```
x_train_01 = x_train[np.logical_or(y_train==0, y_train==1), :, :]
y_train_01 = y_train[np.logical_or(y_train==0, y_train==1)]
x_test_01 = x_test[np.logical_or(y_test==0, y_test==1), :, :]
y_test_01 = y_test[np.logical_or(y_test==0, y_test==1)]
```

[20]:
```
print(x_train_01.shape, y_train_01.shape)
```

```
(12665, 28, 28) (12665,)
```

[21]:
```
print(x_test_01.shape, y_test_01.shape)
```

```
(2115, 28, 28) (2115,)
```

[22]:
```
# Shuffle the training set indices
num_train_01 = x_train_01.shape[0]
train01_ind = np.arange(0, num_train_01)
train01_ind_shuffled = np.random.permutation(train01_ind)

x_train_01 = x_train_01[train01_ind_shuffled,:,:]
y_train_01 = y_train_01[train01_ind_shuffled]

# Select 500 images for validation set
x_valid_01 = x_train_01[0:500, :, :]
y_valid_01 = y_train_01[0:500]
```

```
# The rest of the training set
x_train_01 = x_train_01[500:, :, :]
y_train_01 = y_train_01[500:]
```

[23]:
```
x_train_01.shape
```

[23]:
```
(12165, 28, 28)
```

[24]:
```
features_train = np.sum(x_train_01[:, 13:16, 13:16], axis=2)
```

[25]:
```
features_train.shape
```

[25]:
```
(12165, 3)
```

[26]:
```
features_train = np.sum(features_train, axis=1) / 9
```

[27]:
```
features_train.shape
```

[27]:
```
(12165,)
```

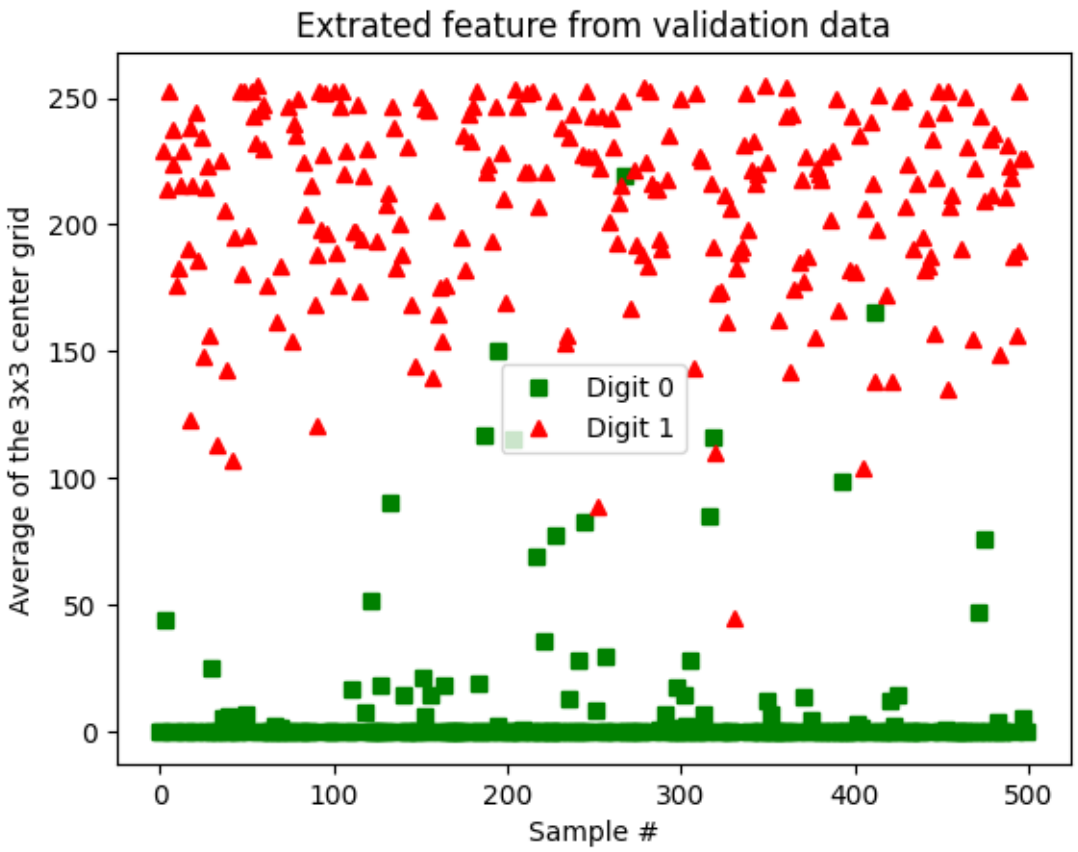
[28]:
```
features_train
```

[28]:
```
array([  0.        ,   0.        , 212.66666667, …, 197.77777778,
         0.        ,   0.        ])
```

[29]:
```
features_val = np.sum(x_valid_01[:, 13:16, 13:16], axis=2)
features_val = np.sum(features_val, axis=1) / 9
```

[30]:
```
features_test = np.sum(x_test_01[:, 13:16, 13:16], axis=2)
features_test = np.sum(features_test, axis=1) / 9
```

[31]:
```
feat_plt(features_val, y_valid_01, [0, 1])
```



[32]:
```
# Time to guess a threshold and calculate the accuracies

y_train_01_pred = np.zeros(features_train.shape)
y_val_01_pred = np.zeros(features_val.shape)
y_test_01_pred = np.zeros(features_test.shape)

threshold = 100

y_train_01_pred[features_train > threshold] = 1
y_train_01_pred[features_train <= threshold] = 0

y_val_01_pred[features_val > threshold] = 1
y_val_01_pred[features_val <= threshold] = 0

y_test_01_pred[features_test > threshold] = 1
y_test_01_pred[features_test <= threshold] = 0

print("Training accuracy:", acc_fun(y_train_01, y_train_01_pred))
print("Validation accuracy:", acc_fun(y_valid_01, y_val_01_pred))
```

```
print("Testing accuracy:", acc_fun(y_test_01, y_test_01_pred))
```

```
Training accuracy: 98.38060008220305
Validation accuracy: 98.4
Testing accuracy: 99.10165484633569
```