CSE 151A - Homework 1
Ashwin Tyagi. A16228593

Problem 1. *Casting an image into vector form.*

Solution.

$$10 \times 10 = 100$$
$$d = 100$$

Problem 2. *Euclidean Distance*

Solution.

$$(1,2,3),(3,2,1)$$
$$dist = \sqrt{(1-3)^2 + (2-2)^2 + (3-1)^2}$$
$$dist = \sqrt{(-2)^2 + (0)^2 + (2)^2}$$
$$dist = \sqrt{4+4}$$
$$dist = \mathbf{2\sqrt{2}}$$

Problem 3. *Accuracy of a Random Classifier*

Solution.

(a) Error rate of a classifier that picks (A,B,C,D) at random.

$$p(any\ letter) = \frac{1}{4}$$

If accuracy is $\frac{1}{4}$ then error rate is $1 - \frac{1}{4} = \mathbf{0.75}$

(b) If we return the same label

- We should return **A** since it has the highest frequency.
- The error rate will be **50%**.

Problem 4. *Decision Boundary of the Nearest Neighbor Classifier*

Solution.

(a) $\mathcal{X} = (0.5, 0.5) \longrightarrow \boldsymbol{\mathcal{Y} = 2}$
After training with.. $X1 : (0.5, 0.5) = 2, X2 : (0.5, 1.5) = 1$

(b)
$$||x - X1|| = \sqrt{(1.5 - 0.5)^2 + (0.5 - 0.5)^2}$$
$$||x - X1|| = \sqrt{(1)^2}$$
$$||x - X1|| = 1$$

$$||x - X2|| = \sqrt{(1.5 - 0.5)^2 + (0.5 - 1.5)^2}$$
$$||x - X2|| = \sqrt{(1)^2 + (1)^2}$$
$$||x - X2|| = \sqrt{2}$$

Since the point (1.5,0.5) is closest to the point (0.5, 0.5)

$$\mathcal{X} = (1.5, 0.5) \longrightarrow \boldsymbol{\mathcal{Y} = 2}$$

(c) Since the point is closest to the point (0.5, 1.5)

$$||x - X1|| = \sqrt{(2 - 0.5)^2 + (2 - 0.5)^2}$$
$$||x - X1|| = \sqrt{(1.5)^2 + (1.5)^2}$$
$$||x - X1|| = \sqrt{2.25 + 2.25}$$
$$||x - X1|| = \sqrt{4.5}$$

$$||x - X2|| = \sqrt{(2 - 0.5)^2 + (2 - 1.5)^2}$$
$$||x - X2|| = \sqrt{(1.5)^2 + (0.5)^2}$$
$$||x - X2|| = \sqrt{2.25 + 0.25}$$
$$||x - X2|| = \sqrt{2.5}$$

Since the point (2,2) is closest to the point (0.5, 1.5)

$$\mathcal{X} = (2, 2) \longrightarrow \boldsymbol{\mathcal{Y} = 1}$$

(d) Since the training set only covers labels for 1 and 2, it will never predict **3**.

(e) Since the training set will never predict $\mathcal{Y} = 3$, any test point in class 3 will never give the right answer. Therefore, the error rate will be **50%**.

Problem 5. *Squares or Stars*

Solution. Let the Query Point be $x = (3.5, 4.5)$.

(a) In 1-NN: We are closest to the point $(4, 4)$ so the point will be classified as a **star**.

$$||x - (4,4)|| = \sqrt{(3.5-4)^2 + (4.5-4)^2}$$
$$||x - (4,4)|| = \sqrt{(0.5)^2 + (0.5)^2}$$
$$||x - (4,4)|| = \sqrt{0.25 + 0.25}$$
$$||x - (4,4)|| = \sqrt{0.5}$$

(b) In 3-NN: We check the closest 3 points $(4, 4), (4, 6), (2, 4)$. Since two of these are squares and only one is a star the point will be classified as a **square**.

$$||x - (4,6)|| = \sqrt{(3.5-4)^2 + (4.5-6)^2}$$
$$||x - (4,6)|| = \sqrt{(-0.5)^2 + (1.5)^2}$$
$$||x - (4,6)|| = \sqrt{0.25 + 2.25}$$
$$||x - (4,6)|| = \sqrt{2.5}$$
$$||x - (2,4)|| = \sqrt{(3.5-2)^2 + (4.5-4)^2}$$
$$||x - (2,4)|| = \sqrt{(-1.5)^2 + (0.5)^2}$$
$$||x - (2,4)|| = \sqrt{2.25 + 0.25}$$
$$||x - (2,4)|| = \sqrt{2.5}$$

(c) In 5-NN: We check the closest 5 points. $(4, 4), (4, 6), (2, 4), (2, 6)$. There are two points that have the same distance to our x point, $(4, 2), (6, 4)$. However, since these are both squares, it doesn't matter which we choose. We get three squares and two stars so the point will be classified as a **square**.

$$||x - (2,6)|| = \sqrt{(3.5-2)^2 + (4.5-6)^2}$$
$$||x - (2,6)|| = \sqrt{(1.5)^2 + (1.5)^2}$$
$$||x - (2,6)|| = \sqrt{2.25 + 2.25}$$
$$||x - (2,6)|| = \sqrt{4.5}$$
$$||x - (4,2)|| = \sqrt{(3.5-4)^2 + (4.5-2)^2}$$
$$||x - (4,2)|| = \sqrt{(-0.5)^2 + (2.5)^2}$$
$$||x - (4,2)|| = \sqrt{0.25 + 6.25}$$
$$||x - (4,2)|| = \sqrt{6.5}$$
$$||x - (6,4)|| = \sqrt{(3.5-6)^2 + (4.5-4)^2}$$
$$||x - (6,4)|| = \sqrt{(-2.5)^2 + (0.5)^2}$$
$$||x - (6,4)|| = \sqrt{6.25 + 0.25}$$
$$||x - (6,4)|| = \sqrt{6.5}$$

Problem 6. *4-fold cross-validation*

Solution. We are doing 4-fold validation to figure out the right value of k on a data set of 10,000. Since it is 4-fold, we split the training set into 4 equal parts.

$$10,000/4 = 2,500$$

Each training set has **2,500** data points.

Problem 7. *Leave-one-out cross-validation*

Solution. Let's start by numbering the points 1,2,3,4.

- LOOCV error for 1-NN.
  test $= 1(+)$ nearest neighbor $= 2(+)$ good
  test $= 2(+)$ nearest neighbor $= 1(+)$ good
  test $= 3(-)$ nearest neighbor $= 4(+)$ wrong
  test $= 4(+)$ nearest neighbor $= 3(-)$ wrong
  There are two errors so error rate $=$ **50%**.

- LOOCV error for 3-NN.
  test $= 1(+)$ nearest neighbor $= 2(+), 3(-), 4(+) = (+)$ good
  test $= 2(+)$ nearest neighbor $= 1(+), 3(-), 4(+) = (+)$ good
  test $= 3(-)$ nearest neighbor $= 4(+), 2(+), 1(+) = (+)$ wrong
  test $= 4(+)$ nearest neighbor $= 3(-), 2(+), 1(+) = (+)$ good
  There is only one errors so error rate $=$ **25%**.

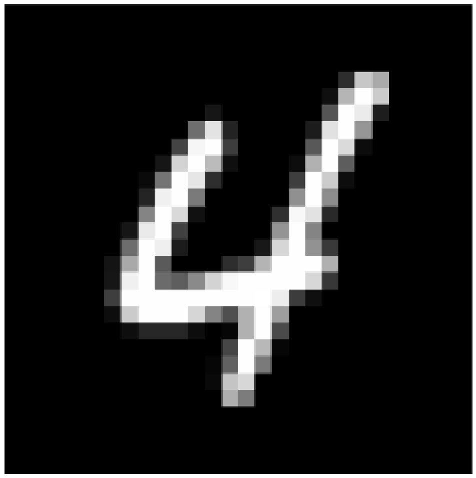
Problem 8. *Nearest Neighbor on MNIST*

Solution.

(a) Test Point 100

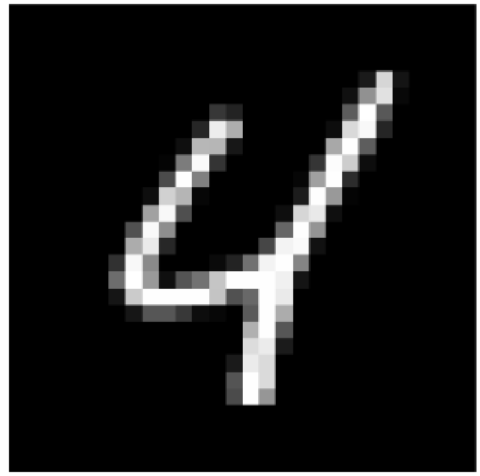
```
# 8a
print("Image 100:")
vis_image(100, "test")
print("Closest neighbor for img 100")
vis_image(find_NN(test_data[100,]), "train")
```

Image 100:



Label 4

Closest neighbor for img 100



Label 4

(b) Confusion Matrix

```
[35]: #8b
from sklearn.metrics import confusion_matrix

def NN_test_label(x):
    label = find_NN(x)
    return train_labels[label]

true_test = test_labels
pred_test = list(map(NN_test_label, test_data))

confusion_matrix = confusion_matrix(true_test, pred_test)
print("L:  0    1    2    3    4    5    6    7    8    9")
print(confusion_matrix)
```

```
L:  0    1    2    3    4    5    6    7    8    9
[[ 99   0   0   0   0   1   0   0   0   0]
 [  0 100   0   0   0   0   0   0   0   0]
 [  0   1  94   1   0   0   0   3   1   0]
 [  0   0   2  91   2   4   0   0   1   0]
 [  0   0   0   0  97   0   0   0   0   3]
 [  1   0   0   0   0  98   0   0   0   1]
 [  0   0   0   0   0   1  99   0   0   0]
 [  0   4   0   0   1   0   0  94   0   1]
 [  2   0   1   1   1   0   1   1  92   1]
 [  1   1   1   1   2   1   0   3   0  90]]
```

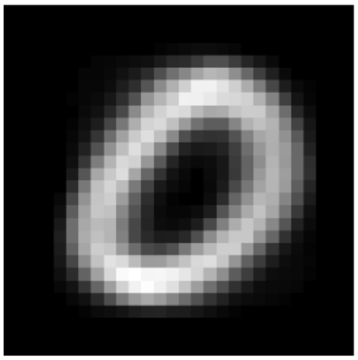
The most misclassified digit was **3**. The least was **1**.

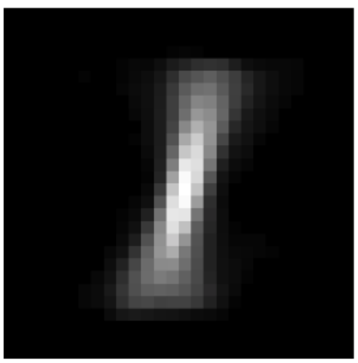
(c) Average Values

[24]:

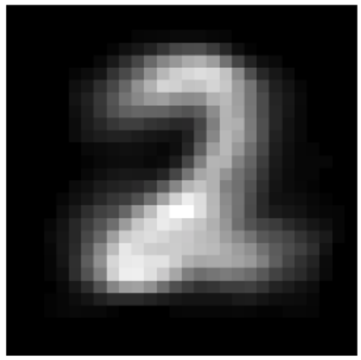
```
#8c code
nn_indicies = list(map(find_NN, test_data))
avg_list = list(range(10))
for i in nn_indicies:
    l = train_labels[i]
    avg_list[l] += train_data[i]

for num in avg_list:
    avg_num = [x//100 for x in num]
    num = avg_num
```

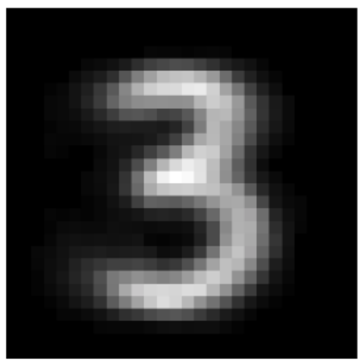
```
show_digit(avg_list[0])
show_digit(avg_list[1])
```



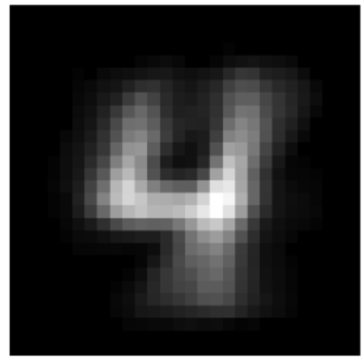


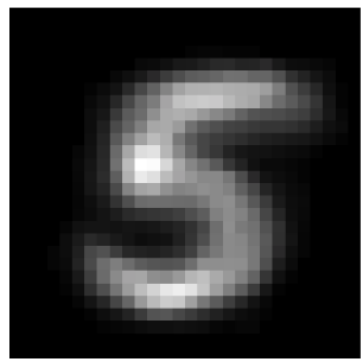
```
show_digit(avg_list[2])
show_digit(avg_list[3])
```



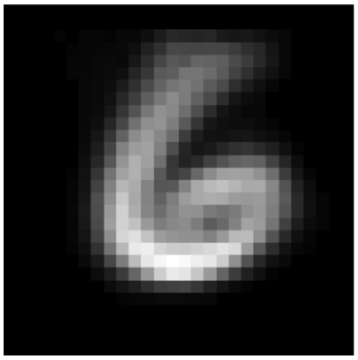


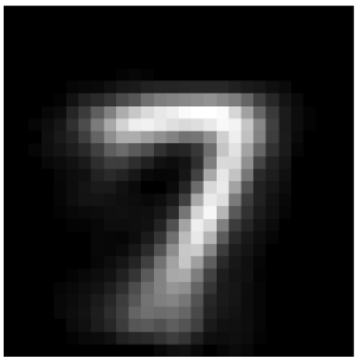
```
show_digit(avg_list[4])
show_digit(avg_list[5])
```



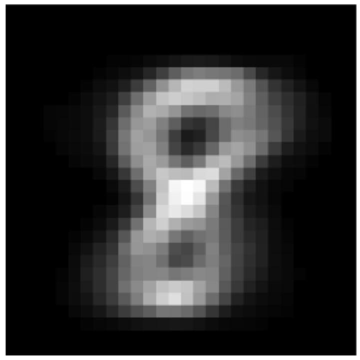


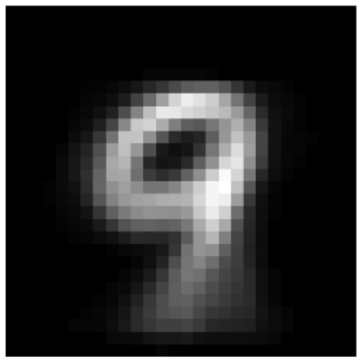
```
show_digit(avg_list[6])
show_digit(avg_list[7])
```





```
show_digit(avg_list[8])
show_digit(avg_list[9])
```





Problem 9. *Back Problems*

Solution. Code:

```
[118]: # Load data set and code labels as 0 = 'NO', 1 = 'DH', 2 = 'SL'
labels = [b'NO', b'DH', b'SL']

data = np.loadtxt('spine-data.txt', converters={6: lambda s: labels.index(s)})

test_BP_raw = data[250:]
training_BP_raw = data[:250]

test_BP_labels = [inner[-1:] for inner in test_BP_raw]
test_BP_data = [inner[:-1] for inner in test_BP_raw]

train_BP_labels = [inner[-1:] for inner in training_BP_raw]
train_BP_data = [inner[:-1] for inner in training_BP_raw]
```

```
[119]: def squared_dist(x,y):
    return np.sum(np.square(x-y))

def lp1(x):
    distances = [np.sum(np.abs(x-train_BP_data[i])) for i in range(len(train_BP_labels))]
    return np.argmin(distances)

def lp2(x):
    distances = [squared_dist(x,train_BP_data[i]) for i in range(len(train_BP_labels))]
    return np.argmin(distances)

def lp1_classifier(x):
    index = lp1(x)
    return train_BP_labels[index]

def lp2_classifier(x):
    index = lp2(x)
    return train_BP_labels[index]
```

(a) Error Rates

[120]:
```python
test_predictions = [lp1_classifier(test_BP_data[i]) for i in range(len(test_BP_labels))]

err_positions = np.not_equal(test_predictions, test_BP_labels)
error = float(np.sum(err_positions))/len(test_BP_labels)

print("Error of lp1: ", error)

test_predictions = [lp2_classifier(test_BP_data[i]) for i in range(len(test_BP_labels))]

err_positions = np.not_equal(test_predictions, test_BP_labels)
error = float(np.sum(err_positions))/len(test_BP_labels)

print("Error of lp2: ", error)
```

```
Error of lp1:  0.21666666666666667
Error of lp2:  0.23333333333333334
```

(b) Confusion Matrix

[122]:
```python
from sklearn.metrics import confusion_matrix

pred_lp1_labels = [lp1_classifier(test_BP_data[i]) for i in range(len(test_BP_labels))]
pred_lp2_labels = [lp2_classifier(test_BP_data[i]) for i in range(len(test_BP_labels))]

confusion_matrix_lp1 = confusion_matrix(test_BP_labels, pred_lp1_labels)
confusion_matrix_lp2 = confusion_matrix(test_BP_labels, pred_lp2_labels)

print(confusion_matrix_lp1)
print(confusion_matrix_lp2)
```

```
[[14  0  2]
 [ 9  9  0]
 [ 1  1 24]]
[[12  1  3]
 [ 9  9  0]
 [ 1  0 25]]
```