Homework 1

Yixuan Li
CSE 151A: Machine Learning and Algorithms

April 10, 2025

1 Conceptual and Mathematical Problems

1. *Costing an image in vector form.*
   (a) $d = 15 \times 15 \times 1 = 225$
   (b) $d = 15 \times 15 \times 3 = 675$

2. *Euclidean distance.*
   $L_2 = \sqrt{(3-1)^2 + (2-(-2))^2 + (1-1)^2} = 2\sqrt{5}$

3. *Accuracy of a random classifier.*
   (a) $\epsilon_1 = 0.5 \times (1 - \frac{1}{4}) + 0.2 \times (1 - \frac{1}{4}) + 0.2 \times (1 - \frac{1}{4}) + 0.1 \times (1 - \frac{1}{4}) = 0.75$

   (b)

   - To obtain the smallest error rate for this classifier, the returned label should be A, since the frequency of label A appears to be the largest in the data set.

   - $\epsilon_2 = 0.5 \times 1 + 0.5 \times 0 = 0.5$

4. *Decision boundary of the nearest neighbor classifier.*
   (a) As in the given plot, the point $m = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}$ is in class 2, so $y_m = 2$.
   (b) Let point $m = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}$, point $n = \begin{bmatrix} 0.5 \\ 1.5 \end{bmatrix}$, point $s = \begin{bmatrix} 1.5 \\ 0.5 \end{bmatrix}$. As in the plot, $y_m = 2, y_n = 1$.
   Compute:
   $$L_1 = \sqrt{(1.5-0.5)^2 + (0.5-0.5)^2} = 1$$
   $$L_2 = \sqrt{(1.5-0.5)^2 + (0.5-1.5)^2} = \sqrt{2}$$
   Since $L_1 < L_2$, the predicted label $y_s = y_m = 2$.
   (c) Let point $t = \begin{bmatrix} 2 \\ 2 \end{bmatrix}$. Compute:
   $$L_3 = \sqrt{(2-0.5)^2 + (2-0.5)^2} = \frac{3\sqrt{2}}{2}$$

$$L_4 = \sqrt{(2-0.5)^2 + (2-1.5)^2} = \frac{\sqrt{10}}{2}$$

Since $L_4 < L_3$, the predicted label $y_t = y_n = 1$.

(d) The classifier will never predict label 3. Because there is no training data point in Class 3.

(e) 50%. Because by 1-NN, the test data point $\begin{bmatrix} a \\ b \end{bmatrix}$ with $a \in [0,1], b \in [0,2]$ can be correctly classified, but the test data point $\begin{bmatrix} c \\ d \end{bmatrix}$ with $c \in [1,2], d \in [0,2]$ cannot be correctly classified. And either of them takes half of the data space.

5. (a) In the plot, the star $\begin{bmatrix} 4 \\ 4 \end{bmatrix}$ is closest to the point $\begin{bmatrix} 3.5 \\ 4.5 \end{bmatrix}$ in Euclidean distance, so the predicted label will be the star by the 1-NN algorithm.

(b) In the plot, the star $\begin{bmatrix} 4 \\ 4 \end{bmatrix}$, the square $\begin{bmatrix} 2 \\ 4 \end{bmatrix}$ and the square $\begin{bmatrix} 4 \\ 6 \end{bmatrix}$ are the three closest points to the point $\begin{bmatrix} 3.5 \\ 4.5 \end{bmatrix}$ in Euclidean distance. Since there are two squares and a star, the predicted label will be the square by the 3-NN algorithm.

(c) In the plot, the star $\begin{bmatrix} 4 \\ 4 \end{bmatrix}$, the square $\begin{bmatrix} 2 \\ 4 \end{bmatrix}$, the square $\begin{bmatrix} 4 \\ 6 \end{bmatrix}$, the square $\begin{bmatrix} 4 \\ 2 \end{bmatrix}$, and the square $\begin{bmatrix} 6 \\ 4 \end{bmatrix}$ are the five closest points to the point $\begin{bmatrix} 3.5 \\ 4.5 \end{bmatrix}$ in the Euclidean distance. Since there are four squares and a star, the predicted label will be the square by the 5-NN algorithm.

6. Every time we take $\frac{1}{4}$ of the data set to be testing set and $\frac{3}{4}$ of the data set to be training set, so the size of each of these training sets is:

$$10000 \times \frac{3}{4} = 7500$$

7. (a) For 1-NN, we can correctly classify the two left "+" data, while misclassify the "-" and "+" data on the right. $\hat{\epsilon}_1 = \frac{0+0+1+1}{4} = 0.5$

(b) For 3-NN, we can correctly classify all the "+" data, and will misclassify the "-" data. $\hat{\epsilon}_2 = \frac{0+0+1+0}{4} = 0.25$

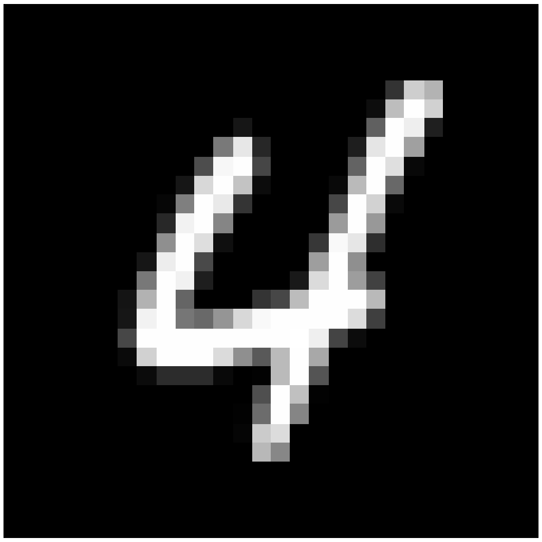
8. It takes time $O(n^2 d)$ to estimate the error of this classifier using LOOCV.

2 Programming Problems

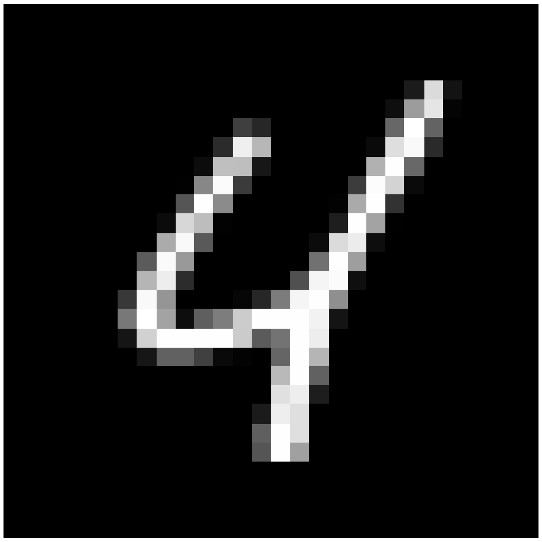
9. (a) By using the following code, we can print the test point #100 and its nearest neighbor in the training set:

```
## Test point 100
print("Test point 100:")
print("NN classification: ", NN_classifier(test_data[100,]))
print("True label: ", test_labels[100])
print("The test image:")
vis_image(100, "test")
print("The corresponding nearest neighbor image:")
vis_image(find_NN(test_data[100,]), "train")

nearest_index = find_NN(test_data[100,])
print(f"The index of its nearest neighbor in the training set {
                                    nearest_index}")
```

The two images are as follows:



(a) Test point #100: image



(b) Test point #100: nearest neighbor

Figure 1: Problem 9 (a)

Based on the output label and the image, we can see that this test point is classified correctly.

(b) To compute the confusion matrix, we use the code:

```
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay

# Predict all the labels in test dataset
print("Running 1-NN on test set...")
y_prediction = [NN_classifier(x) for x in test_data]

# Construct the confusion matrix
N = confusion_matrix(test_labels, y_prediction)

# Visualize the confusion matrix
```

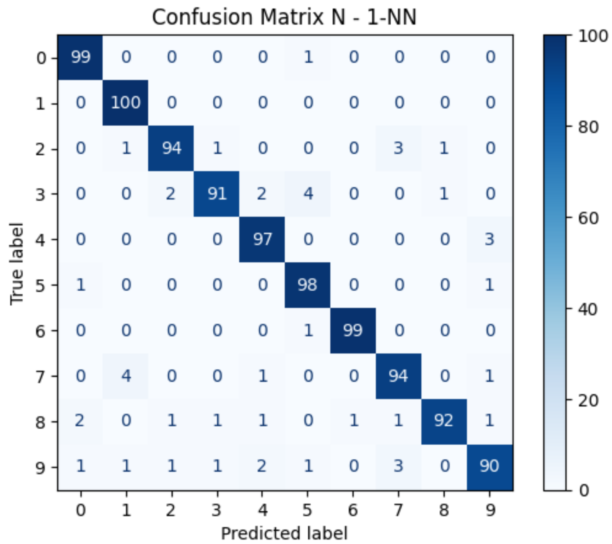


Figure 2: Problem 9 (b) Confusion Matrix for the 1-NN

```
disp = ConfusionMatrixDisplay(confusion_matrix=N, display_labels=range(10)
                                          )
disp.plot(cmap=plt.cm.Blues)
plt.title("Confusion Matrix N - 1-NN")
plt.show()
```

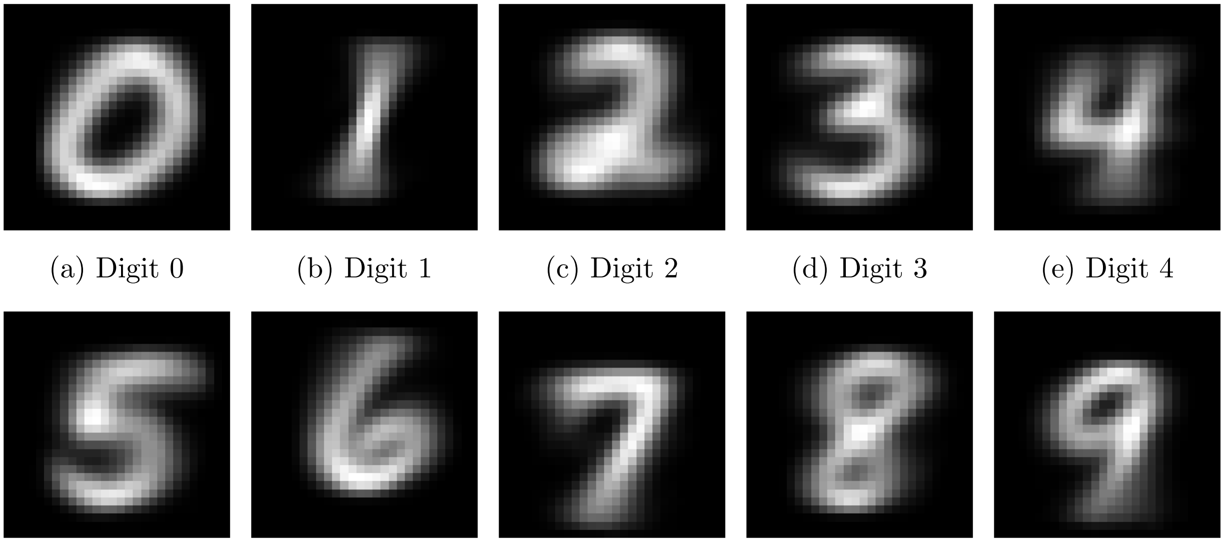
Then we can get the image of the confusion matrix:

Based on the image, the digit 9 is misclassified the most often (correct: 90/100), and the digit 1 is misclassified the least often (correct: 100/100).

(c) For each digit $0 \leq i \leq 9$, we compute the mean of all training instances of image i, and use the **show_digit** routine to print out these 10 average-digits. The code is as follows.

```
for i in range(0,10):
    mean = train_data[train_labels == i].mean(axis=0)
    show_digit(mean)
    print("Average Digits " + str(i))
```

And the output images for each digit $0 \leq i \leq 9$ is as follows:



(a) Digit 0 (b) Digit 1 (c) Digit 2 (d) Digit 3 (e) Digit 4

(f) Digit 5 (g) Digit 6 (h) Digit 7 (i) Digit 8 (j) Digit 9

Figure 3: Problem 9 (c) Average digit from 0 to 9

10. Based on the steps in the question, we load the `spine-data.txt`, split the data into a training set and a testing set following the guidance, and calculate the L2 and L1 distance.

(a) We calculate the error rate on the test set for L2 and L1 distance based on the following code:

```
l2_test_predictions = [l2_NN_classifier(test_data[i,]) for i in range(len(
                                            test_labels))]
l1_test_predictions = [l1_NN_classifier(test_data[i,]) for i in range(len(
                                            test_labels))]
## Compute the error
l2_err_positions = np.not_equal(l2_test_predictions, test_labels)
l2_error = float(np.sum(l2_err_positions))/len(test_labels)

l1_err_positions = np.not_equal(l1_test_predictions, test_labels)
l1_error = float(np.sum(l1_err_positions))/len(test_labels)
```

And the output error rate is:

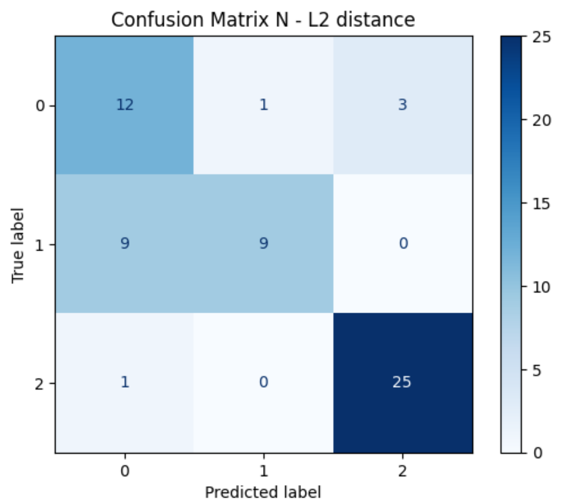
- L2 distance: 23.33%
- L1 distance: 21.67%

(b) Calculate the confusion matrix of the NN classifier for L2 and L1 distance based on the code:

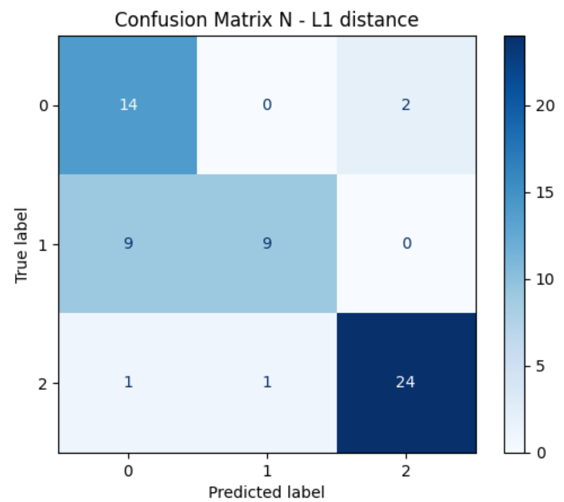
```
# Predict all the labels in test dataset
l2_y_prediction = [l2_NN_classifier(x) for x in test_data]
l1_y_prediction = [l1_NN_classifier(x) for x in test_data]

# Construct the confusion matrix
N_l2 = confusion_matrix(test_labels, l2_y_prediction)
N_l1 = confusion_matrix(test_labels, l1_y_prediction)
```

```python
# Visualize the confusion matrix
disp1 = ConfusionMatrixDisplay(confusion_matrix=N_l2, display_labels=range
                                                (3))
disp1.plot(cmap=plt.cm.Blues)
plt.title("Confusion Matrix N - L2 distance")

disp2 = ConfusionMatrixDisplay(confusion_matrix=N_l1, display_labels=range
                                                (3))
disp2.plot(cmap=plt.cm.Blues)
plt.title("Confusion Matrix N - L1 distance")

plt.show()
```

We can see the displayed confusion matrix:



(a) L2 Distance



(b) L1 Distance

Figure 4: Problem 10 (b) Confusion matrix for L2 and L1 Distance