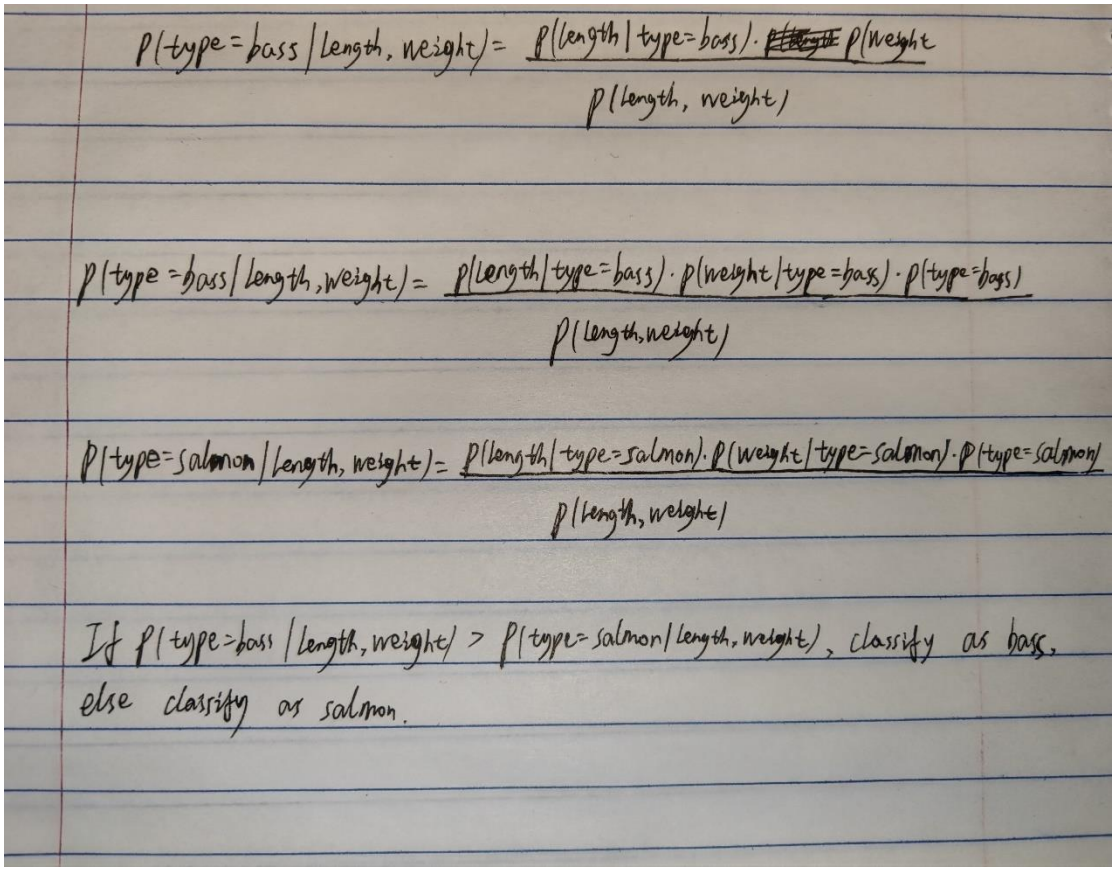
ENGN 2520 Pattern Recognition and Machine Learning

Homework 3

Zhuo Wang

Problem 1

(a)



(b)

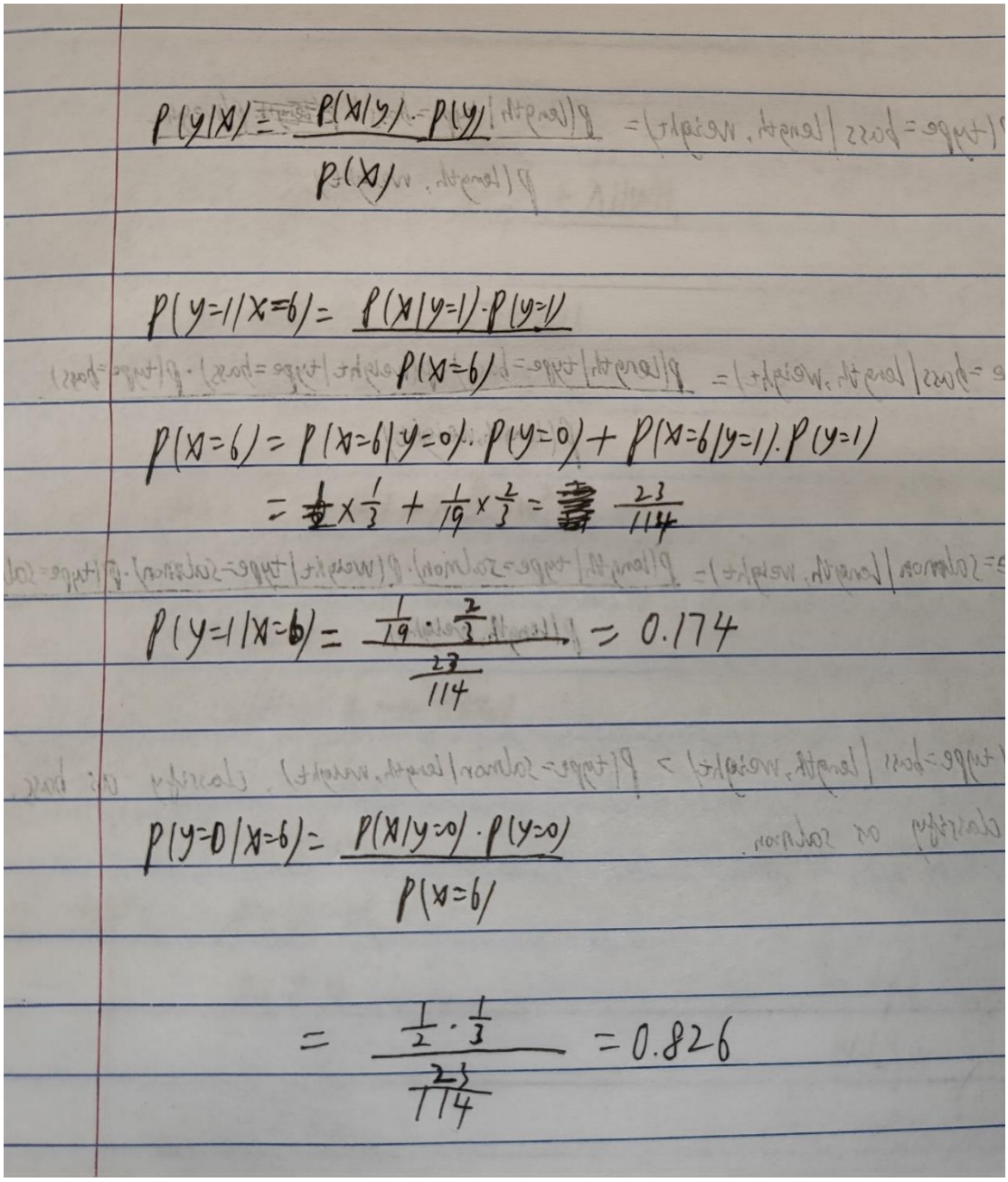
Inadequate training data results in a training set that may not adequately represent the variability present in the overall fish. Cause performance degradation.

The assumption of independence between the length and weight of the fish given the fish type may not hold true in practice. Maybe bass and salmon don't differ much in length or weight. The performance of the classifier may deteriorate.

The classifier's model may be too simplistic to capture the underlying patterns in the data. If the relationship between features and fish type is non-linear, it may be difficult to make accurate predictions.

Problem 2

(a)



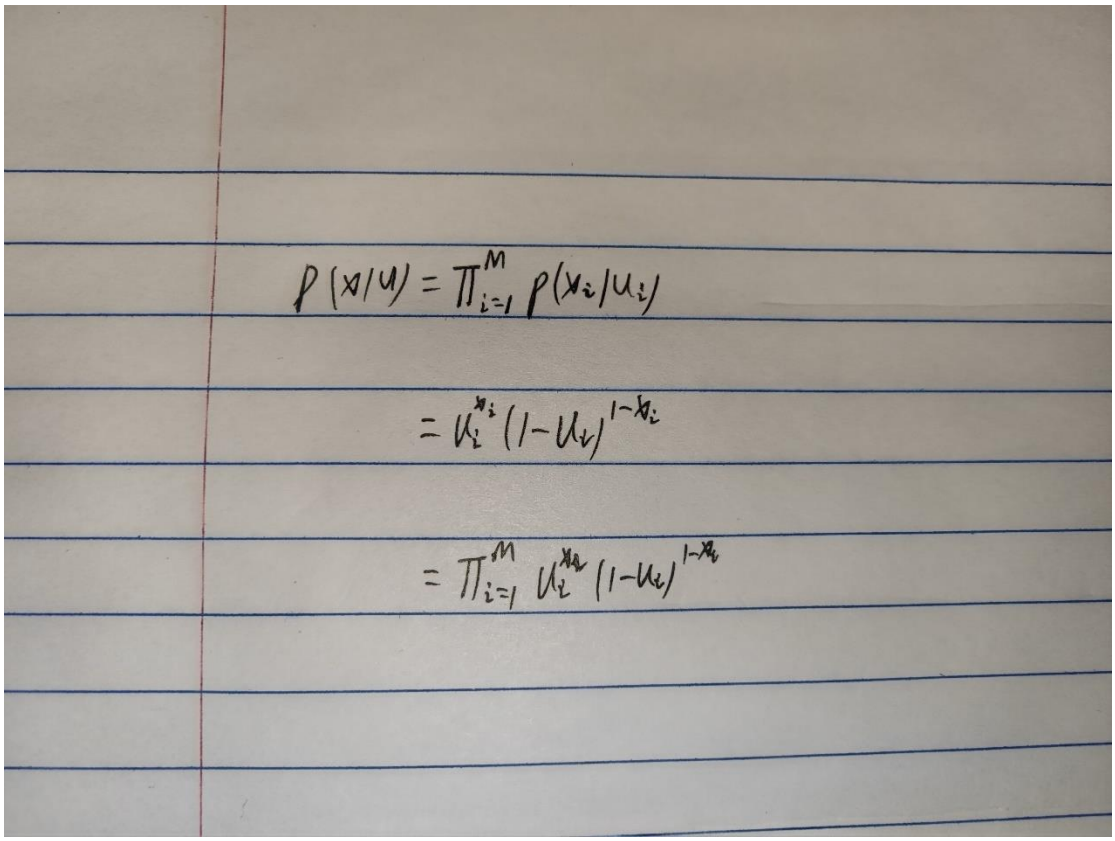
(b)

The Bayes optimal classifier for this example assigns the input x to the class with the higher posterior probability. The Bayes optimal classifier would classify x=6 as belonging to class 0.

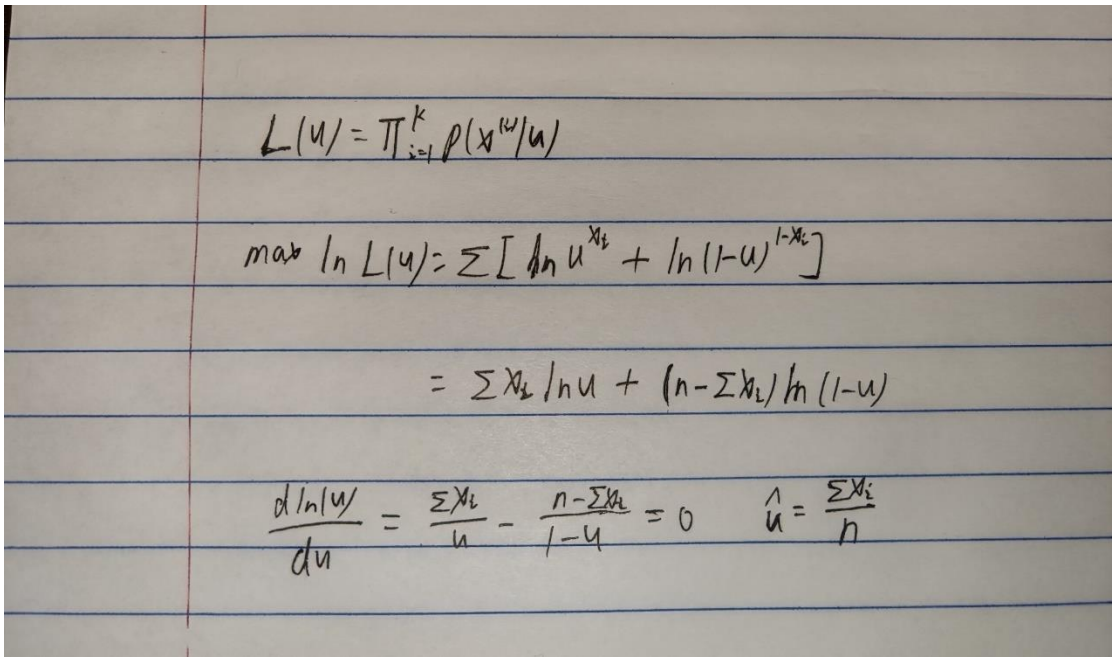
The decision boundary occurs at x=6. The input space is partitioned into two decision regions. For x<6, the classifier assigns y=0. For x>6, the classifier assigns y=1.

Problem 3

(a)

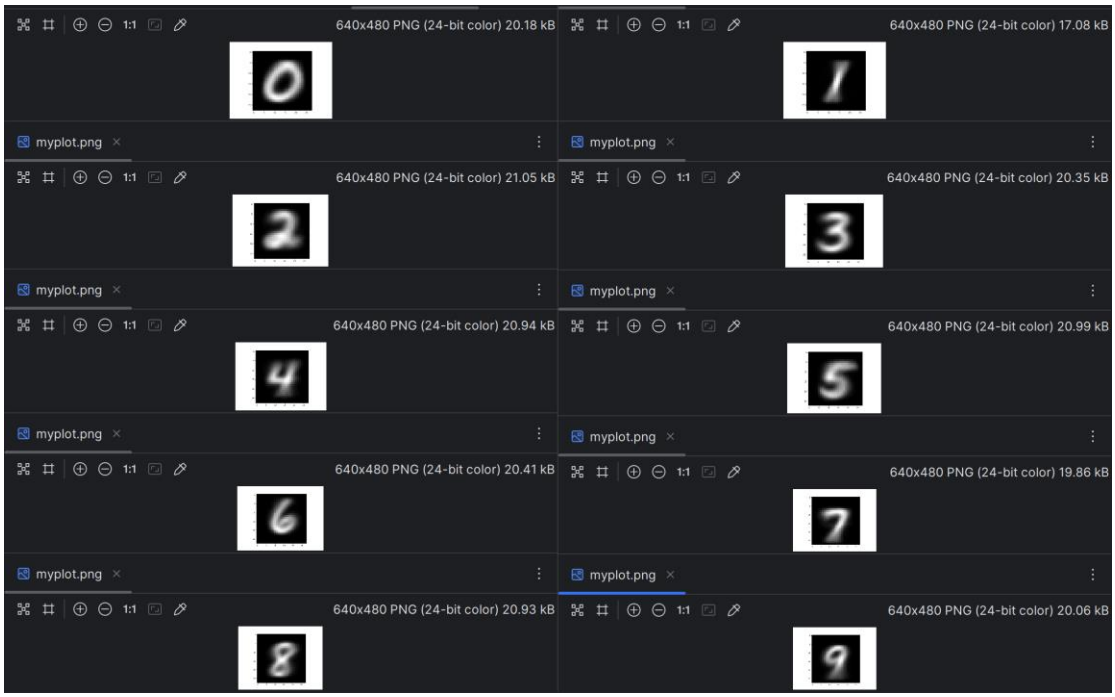

$P(x|u) = \prod_{i=1}^{n} P(x_i|u_i)$
$= u_i^{x_i}(1-u_i)^{1-x_i}$
$= \prod_{i=1}^{n} u_i^{x_i}(1-u_i)^{1-x_i}$

(b)


$L(u) = \prod_{i=1}^{k} P(x^{(i)}|u)$
$\max \ln L(u) = \sum [\ln u^{x_i} + \ln(1-u)^{1-x_i}]$
$= \sum x_i \ln u + (n - \sum x_i)\ln(1-u)$
$\frac{d \ln(u)}{du} = \frac{\sum x_i}{u} - \frac{n-\sum x_i}{1-u} = 0 \quad \hat{u} = \frac{\sum x_i}{n}$

Problem 4

(a)

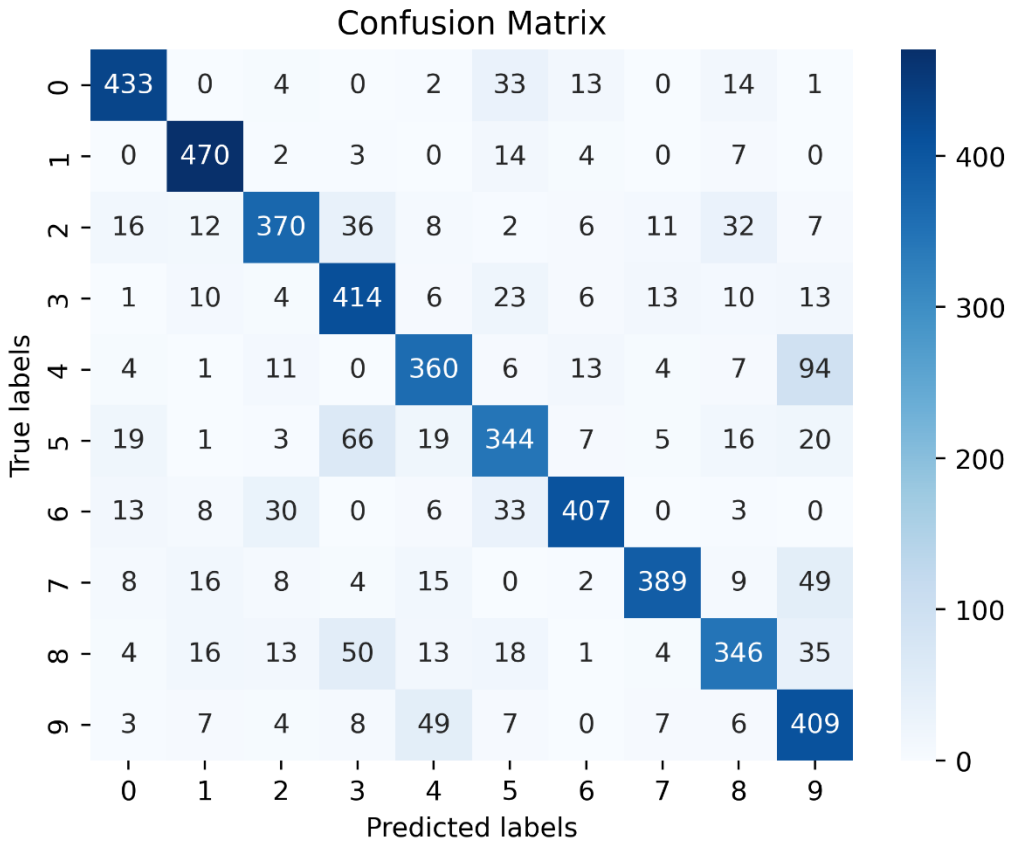


(b)

78.84% of the digits were successfully sorted.

```
C:\Users\25303\anaconda3\python.exe C:\Users\25303\PycharmProjects\pythonProject3\HW3.py
[[433   0   4   0   2  33  13   0  14   1]
 [  0 470   2   3   0  14   4   0   7   0]
 [ 16  12 370  36   8   2   6  11  32   7]
 [  1  10   4 414   6  23   6  13  10  13]
 [  4   1  11   0 360   6  13   4   7  94]
 [ 19   1   3  66  19 344   7   5  16  20]
 [ 13   8  30   0   6  33 407   0   3   0]
 [  8  16   8   4  15   0   2 389   9  49]
 [  4  16  13  50  13  18   1   4 346  35]
 [  3   7   4   8  49   7   0   7   6 409]]
Accuracies for each digits (0-9): [0.866, 0.94, 0.74, 0.828, 0.72, 0.688, 0.814, 0.778, 0.692, 0.818]
Accuracies for total digits: 0.7884

Process finished with exit code 0
```



(c)
The digits 5 and 8 were often misclassified.

Naive Bayes works reasonably for this task. Because the Bernulli distribution of each pixel is independent of each other.

But naive Bayes ignores the relationship between spatial information and adjacent pixels. In handwritten digit recognition, adjacent pixels may be related, especially when capturing curves in a digit. This may be the cause of poor recognition of the digits 5 and 8.

Source Code

```
from scipy.io import loadmat
from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np

def load_data(nums):
    train_data = loadmat('digits.mat')[f'train{nums}']
    test_data = loadmat('digits.mat')[f'test{nums}']
    return train_data, test_data

def visualize(pixel_means):
    model_image = pixel_means.reshape(28, 28)
    plt.imshow(model_image, cmap='gray')
    plt.show()

def classify_example(test_example, ms):
    predictions = []

    for image in test_example:
        probabilities = []
        for model in ms:
            probability = 1.0
            for i in range(len(image)):
                if image[i] == 1:
                    probability *= model[i]
                else:
                    probability *= (1 - model[i])
            probabilities.append(probability)

        predictions.append(np.argmax(probabilities))

    return predictions

def test_classifier(ms):

    pp = []

    for i in range(10):
        _, test_data = load_data(i)
        predicted_label = classify_example(test_data, ms)
        pp += predicted_label
```

```
    return pp

def train_naive_bayes():
    models = []
    for i in range(10):
        train_data, _ = load_data(i)
        pixel_means = np.mean(train_data, axis=0)
        models.append(pixel_means)
        #visualize(pixel_means)
    return models

m = train_naive_bayes()

sequence = [i for i in range(10) for _ in range(500)]

te = test_classifier(m)

conf_mat = confusion_matrix(sequence, te)

print(conf_mat)

sns.heatmap(conf_mat, cmap='Blues', annot=True, fmt='d', cbar=True)
plt.title('Confusion Matrix')
plt.xlabel('Predicted labels')
plt.ylabel('True labels')
plt.savefig('cf.png', dpi=400, bbox_inches='tight')
plt.show()


accuracies = []
for i in range(conf_mat.shape[0]):
    correct_predictions = conf_mat[i, i]
    total_samples = np.sum(conf_mat[i, :])
    accuracy = correct_predictions / total_samples
    accuracies.append(accuracy)

print("Accuracies for each digits (0-9):", accuracies)
print("Accuracies for total digits:", np.mean(accuracies))
```