

## Spring 2024 Introduction to Deep Learning

### Homework Assignment 1 Due dates: Feb. 15

**Problem 1 (Practice the computation of KNN).** You are required to use KNN to classify a 3-dimension data. The training dataset contains 12 pairs of (data, label) as follows:

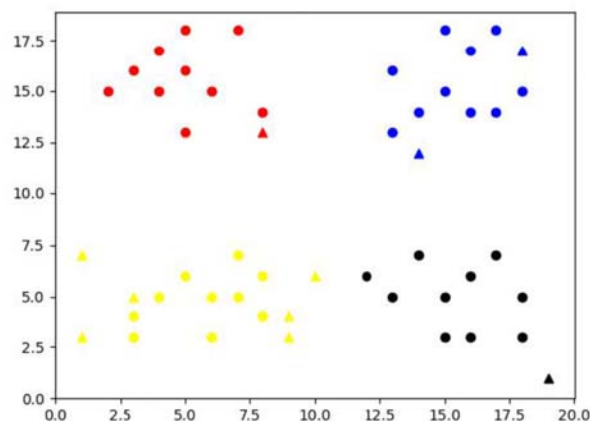
Class A: (0,1,0), (0, 1,1), (1,2,1),(1,2,0)

Class B: (1,2,2),(2,2,2),(1,2,-1),(2,2,3)

Class C: (-1,-1,-1),(0,-1,-2),(0,-1,1),(-1,-2,1)

What is classified label for test data (1,0,1) when  $K=1, 2$ , and  $3$ , respectively? Choose L2 distance as the measurement metric.

**Problem 2 (KNN for simple data).** There are 40 2-dimension training data and corresponding labels (0~3) have been saved in the “knn\_minitrain.npy” and “knn\_minitrain\_label.npy”. Write a KNN classifier with filename “miniknn.py” to classify 10 random generated 2-dimension test data. Visualized result is illustrated as follows, where round and triangle indicate train and test data, respectively. The value of  $k$  can be chosen between 3~10.



**Note:** Part of “miniknn.py” for data loading and plotting has been given. You can utilize them or write your own codes. Also, choose L2 distance as the measurement metric.

**Problem 3 (KNN for handwriting digit recognition).** In this problem you will use KNN to recognize handwritten digits.

First, use “download\_mnist.py” file to download the MNIST database. This file will make data to following numpy arrays and save it as Pickle. (“mnist.pkl”)

x\_train : 60,000x784 numpy array that each row contains flattened version of training images.

y\_train : 1x60,000 numpy array that each component is true label of the corresponding training images.

x\_test : 10,000x784 numpy array that each row contains flattened version of test images.

y\_test : 1x10,000 numpy array that each component is true label of the corresponding test images.

**Notice:** Once you get “mnist.pkl”, you don't need to call init() anymore. Everything you need to do is to locate “download\_mnist.py” and “mnist.pkl” in your working directory and to call load(). Then you can load the MNIST database in “knn.py”

**Notice:** Due to the high computational complexity of KNN, you do not need to classify all 10000 test images. Instead, you can select how many test images to classify in line51 and line52 of “knn.py” (e.g. 20 images).

**Note:** Part of “knn.py” for data loading and plotting has been given. You can utilize them or write your own codes. You can choose the value of  $k$ . Also, choose L2 distance as the measurement metric.

The final accuracy should be over 95% depending on your test dataset.

Execution time may be different due to the hardware platform and size of test data.

```
---classification accuracy for knn on mnist: 1.0 ---  
---execution time: 8.206248044967651 seconds ---
```

### What to submit:

Submit a PDF-version report for three problems, including source codes and print screen results.