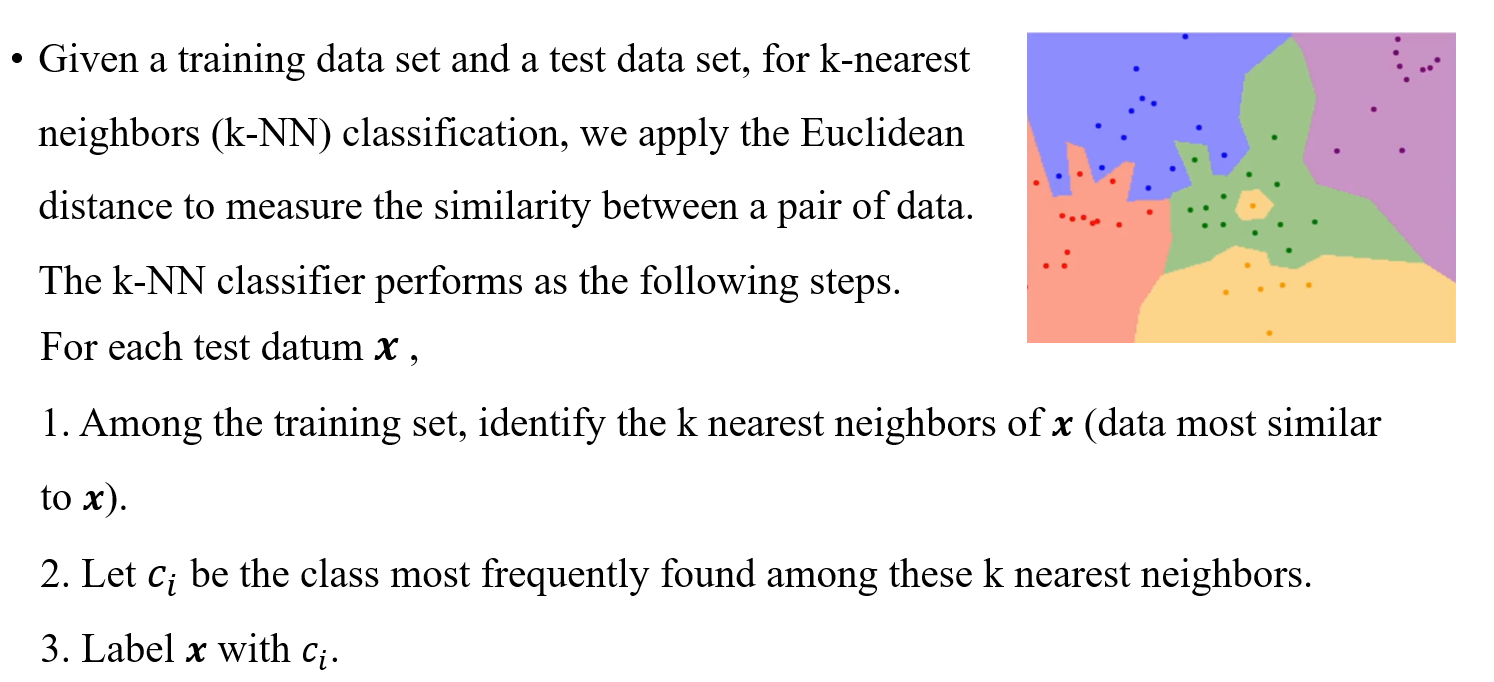
**《深度学习理论与实践》实验报告**

**姓 名：** 徐云哲  **学 号：** 21200211047

**年 级：** 20级 **专 业：** 计算机软件与理论

1. **实验名称**

**Experiment 3 K-Nearest Neighbors for Classification**

1. **实验要求**

Download a classification dataset from the UCI machine learning repository（https://archive.ics.uci.edu/ml/datasets.php）.

Split the datdaset into a training set and a test set.

Write the Python code of k-nearest neighbors to solve the classification problem.

1. **实验过程**

**1、数据集介绍：**

Iris数据集一共包括150行记录，其中前四列为花萼长度，花萼宽度，花瓣长度，花瓣宽度等4个用于识别鸢尾花的属性，第5列为鸢尾花的类别（包括Setosa，Versicolour，Virginica三类）。也即通过判定花萼长度，花萼宽度，花瓣长度，花瓣宽度的尺寸大小来识别鸢尾花的类别。

（注：在实验中只取数据集的前100个数据的前两个属性进行实验）

**2、数据集处理：**

（1）数据获取

# 数据获取

def get\_data():

    iris = load\_iris()

    data = iris.data

    result = iris.target

    return data, result

（2）将数据集分为训练集和测试集

# 将数据处理为训练集和测试集

def data\_deal(data, result):

    data\_list = []

    for i in data:

        tem\_list = [i[0], i[1]]

        data\_list.append(tem\_list)

    res\_list = []

    for j in result:

        res\_list.append(j)

    train\_list = data\_list[0: 10] + data\_list[20: 80] + data\_list[90: 100]

    train\_result = res\_list[0: 10] + res\_list[20: 80] + res\_list[90: 100]

    test\_list = data\_list[0: 40] + data\_list[60: 100]

    test\_result = res\_list[0: 40] + res\_list[60: 100]

    return data\_list, train\_list, test\_list, train\_result, test\_result

返回值为训练集、训练结果、测试集、测试集标准结果（用于与测试集最终结果比对计算准确率）

（3）K近邻(KNN)距离计算

# k近邻法计算距离方法

def distance(A, B):

    return (abs((B[0]-A[0])\*\*2+(B[1]-A[1])\*\*2))\*\*0.5

（4）计算准确率

k近邻法思路为：给定一个点，选取与其距离最近的n个点，若这n个点中属于0类的点多，那么可以近似的认为该点也属于0类，否则将该点归为1类。在对所有点进行分类后，与标准结果进行比对，得出准确率。

# k近邻法训练，测试准确率

def K\_train(train\_list, train\_result, k):

    dis\_list = []

    # 所有点到别的点的距离

    for i in train\_list:

        dis = []

        for j in train\_list:

            dis.append(distance(i, j))

        dis\_list.append(dis)

    # 获取到训练集中每个点的最近5个点的索引

    min\_dis\_list = []

    for m in range(len(dis\_list)):

        temp = []

        for n in range(int(k+1)):

            temp.append(dis\_list[m].index(min(dis\_list[m])))

            dis\_list[m][dis\_list[m].index(min(dis\_list[m]))] = 100

            temp.sort()

        x = temp[1:]

        min\_dis\_list.append(x)

    # 根据索引判断对应点的类别

    dot\_type = []

    for ii in min\_dis\_list:

        mm = 0

        nn = 0

        for jj in ii:

            if jj <= 50:

                mm = mm+1

            else:

                nn = nn+1

        if mm >= nn:

            dot\_type.append(0)

        else:

            dot\_type.append(1)

    # 计算准确率

    count = 0

    for xx in range(len(dot\_type)):

        if dot\_type[xx] == train\_result[xx]:

            count = count+1

return count/len(dot\_type)

（5）K近邻法判断输入点类别

输入一个点，使用k近邻法判断其所属类别。

# k近邻法判断点的类型

def K\_check\_point(dots, check\_list, result):

    dis = []

    for i in check\_list:

        dis.append(distance(dots, i))

    min\_dis = []

    for j in range(5):

        min\_dis.append(dis.index(min(dis)))

        dis[dis.index(min(dis))] = 100

    zero = 0

    one = 0

    for s in min\_dis:

        if result[s] == 0:

            zero = zero+1

        else:

            one = one+1

    if one > zero:

        return "K近邻法判断该点类别为0"

    else:

        return "K近邻法判断该点类别为1"

（6）绘制函数图像

使用matplotlib进行绘图，一张绘制所有点，另一张对训练集和测试集进行分别绘制，并将用户输入的点绘制到图像上。

# 绘制函数图像和输入点

def fun\_image(w, b, dot):

    iris = load\_iris()

    irisFeature = iris.data

    irisTarget = iris.target

    ax1 = plt.subplot(1, 2, 1)

    ax2 = plt.subplot(1, 2, 2)

    plt.sca(ax1)

    for i in range(0, 100):

        if irisTarget[i] == 0:

            type11 = plt.scatter(irisFeature[i, 0], irisFeature[i, 1], c="r")

        elif irisTarget[i] == 1:

            type22 = plt.scatter(irisFeature[i, 0], irisFeature[i, 1], c="g")

    plt.title("show train and test")

    plt.xlabel("ewidth")

    plt.ylabel("elength")

    x = np.linspace(4, 7, 256)

    plt.plot(x, w \* x + b, color='black')

    plt.legend((type11, type22), ('0', '1'))

    plt.plot(dot[0], dot[1], color='black', marker='+')

    plt.sca(ax2)

    for i in range(20, 80):

        if irisTarget[i] == 0:

            type1 = plt.scatter(irisFeature[i, 0], irisFeature[i, 1], c="r", marker="8")

        elif irisTarget[i] == 1:

            type2 = plt.scatter(irisFeature[i, 0], irisFeature[i, 1], c="g", marker="8")

    for i in range(0, 20):

        if irisTarget[i] == 0:

            type3 = plt.scatter(irisFeature[i, 0], irisFeature[i, 1], c="blue", marker="v")

        elif irisTarget[i] == 1:

            plt.scatter(irisFeature[i, 0], irisFeature[i, 1], c="orange", marker="v")

    for i in range(80, 100):

        if irisTarget[i] == 0:

            plt.scatter(irisFeature[i, 0], irisFeature[i, 1], c="blue", marker="v")

        elif irisTarget[i] == 1:

            type4 = plt.scatter(irisFeature[i, 0], irisFeature[i, 1], c="orange", marker="v")

    plt.title("show all")

    plt.xlabel("ewidth")

    plt.ylabel("elength")

    x = np.linspace(4, 7, 256)

    plt.plot(x, w\*x+b, color='black')

    plt.plot(dot[0], dot[1], color='black', marker='+')

    plt.legend((type1, type2, type3, type4), ('train-0', 'train-1', 'test-0', 'test-1'))

    plt.show()

1. **实验结果**

结合线性回归同时进行实验， **+** 为手动输入的需要判断的点。

