**实验（实习）报告**

1. 实验目的
2. 了解KNN的基本概念；
3. 解如何使用MindSpore进行KNN实验。
4. 实验任务

使用MindSpore在部分wine数据集上进行KNN实验。

1. 实验步骤

1、导入MindSpore模块和辅助模块

import os

# os.environ['DEVICE\_ID'] = '4'

import csv

import numpy as np

import mindspore as ms

from mindspore import context

from mindspore import nn

from mindspore.ops import operations as P

from mindspore.ops import functional as F

2、读取Wine数据集wine.data，并查看部分数据。

with open('wine.data') as csv\_file:

    data = list(csv.reader(csv\_file, delimiter=','))

print(data[56:62]+data[130:133]) # 打印部分数据、

1. 取三类样本（共178条），将数据集的13个属性作为自变量X。将数据集的3个类别作为因变量Y。

X = np.array([[float(x) for x in s[1:]] for s in data[:178]], np.float32)

Y = np.array([s[0] for s in data[:178]], np.int32)

1. 取样本的某两个属性进行2维可视化，可以看到在某两个属性上样本的分布情况以及可分性。

from matplotlib import pyplot as plt

%matplotlib inline

attrs = ['Alcohol', 'Malic acid', 'Ash', 'Alcalinity of ash', 'Magnesium', 'Total phenols',

         'Flavanoids', 'Nonflavanoid phenols', 'Proanthocyanins', 'Color intensity', 'Hue',

         'OD280/OD315 of diluted wines', 'Proline']

plt.figure(figsize=(10, 8))

for i in range(0, 4):

    plt.subplot(2, 2, i+1)

    a1, a2 = 2 \* i, 2 \* i + 1

    plt.scatter(X[:59, a1], X[:59, a2], label='1')

    plt.scatter(X[59:130, a1], X[59:130, a2], label='2')

    plt.scatter(X[130:, a1], X[130:, a2], label='3')

    plt.xlabel(attrs[a1])

    plt.ylabel(attrs[a2])

    plt.legend()

plt.show()

1. 将数据集按128:50划分为训练集（已知类别样本）和验证集（待验证样本）：

train\_idx = np.random.choice(178, 128, replace=False)

test\_idx = np.array(list(set(range(178)) - set(train\_idx)))

X\_train, Y\_train = X[train\_idx], Y[train\_idx]

X\_test, Y\_test = X[test\_idx], Y[test\_idx]

### 计算距离。利用MindSpore提供的tile, suqare, ReduceSum, sqrt, TopK等算子，通过矩阵运算的方式同时计算输入样本x和已明确分类的其他样本X\_train的距离，并计算出top k近邻。

class KnnNet(nn.Cell):

    def \_\_init\_\_(self, k):

        super(KnnNet, self).\_\_init\_\_()

        self.tile = P.Tile()

        self.sum = P.ReduceSum()

        self.topk = P.TopK()

        self.k = k

    def construct(self, x, X\_train):

        # Tile input x to match the number of samples in X\_train

        x\_tile = self.tile(x, (128, 1))

        square\_diff = F.square(x\_tile - X\_train)

        square\_dist = self.sum(square\_diff, 1)

        dist = F.sqrt(square\_dist)

        # -dist mean the bigger the value is, the nearer the samples are

        values, indices = self.topk(-dist, self.k)

        return indices

def knn(knn\_net, x, X\_train, Y\_train):

    x, X\_train = ms.Tensor(x), ms.Tensor(X\_train)

    indices = knn\_net(x, X\_train)

    topk\_cls = [0]\*len(indices.asnumpy())

    for idx in indices.asnumpy():

        topk\_cls[Y\_train[idx]] += 1

    cls = np.argmax(topk\_cls)

    return cls

### 7、预测。在验证集上验证KNN算法的有效性，取k = 5，验证精度接近80%，说明KNN算法在该3分类任务上有效，能根据酒的13种属性判断出酒的品种。

acc = 0

knn\_net = KnnNet(5)

for x, y in zip(X\_test, Y\_test):

    pred = knn(knn\_net, x, X\_train, Y\_train)

    acc += (pred == y)

    print('label: %d, prediction: %s' % (y, pred))

print('Validation accuracy is %f' % (acc/len(Y\_test)))

label: 1, prediction: 1  
label: 1, prediction: 1  
label: 2, prediction: 3  
label: 3, prediction: 2  
label: 1, prediction: 3  
label: 3, prediction: 3  
label: 1, prediction: 1  
label: 1, prediction: 1  
label: 3, prediction: 3  
label: 1, prediction: 1  
label: 3, prediction: 3  
label: 1, prediction: 1  
label: 1, prediction: 1  
label: 1, prediction: 1  
label: 3, prediction: 1  
label: 3, prediction: 2  
label: 1, prediction: 1  
label: 3, prediction: 2  
label: 3, prediction: 3  
label: 3, prediction: 1  
label: 3, prediction: 3  
label: 3, prediction: 3  
label: 3, prediction: 2  
label: 1, prediction: 1  
label: 1, prediction: 1  
label: 3, prediction: 2  
label: 3, prediction: 3  
label: 1, prediction: 1  
label: 3, prediction: 3  
label: 1, prediction: 1  
label: 1, prediction: 1  
label: 1, prediction: 1  
label: 2, prediction: 3  
label: 2, prediction: 1  
label: 2, prediction: 3  
label: 2, prediction: 1  
label: 2, prediction: 2  
label: 2, prediction: 3  
label: 2, prediction: 1  
label: 2, prediction: 3  
label: 2, prediction: 3  
label: 2, prediction: 2  
label: 2, prediction: 2  
label: 2, prediction: 2  
label: 2, prediction: 2  
label: 2, prediction: 3  
label: 2, prediction: 2  
label: 2, prediction: 3  
label: 2, prediction: 2  
label: 2, prediction: 2  
Validation accuracy is 0.620000

1. 实验结果
2. 完整代码

import os

# os.environ['DEVICE\_ID'] = '4'

import csv

import numpy as np

import mindspore as ms

from mindspore import context

from mindspore import nn

from mindspore.ops import operations as P

from mindspore.ops import functional as F

context.set\_context(device\_target="Ascend")

with open('wine.data') as csv\_file:

    data = list(csv.reader(csv\_file, delimiter=','))

print(data[56:62]+data[130:133]) # 打印部分数据

X = np.array([[float(x) for x in s[1:]] for s in data[:178]], np.float32)

Y = np.array([s[0] for s in data[:178]], np.int32)

from matplotlib import pyplot as plt

%matplotlib inline

attrs = ['Alcohol', 'Malic acid', 'Ash', 'Alcalinity of ash', 'Magnesium', 'Total phenols',

         'Flavanoids', 'Nonflavanoid phenols', 'Proanthocyanins', 'Color intensity', 'Hue',

         'OD280/OD315 of diluted wines', 'Proline']

plt.figure(figsize=(10, 8))

for i in range(0, 4):

    plt.subplot(2, 2, i+1)

    a1, a2 = 2 \* i, 2 \* i + 1

    plt.scatter(X[:59, a1], X[:59, a2], label='1')

    plt.scatter(X[59:130, a1], X[59:130, a2], label='2')

    plt.scatter(X[130:, a1], X[130:, a2], label='3')

    plt.xlabel(attrs[a1])

    plt.ylabel(attrs[a2])

    plt.legend()

plt.show()

plt.savefig('result.png',dpi=600, bbox\_inches='tight')

train\_idx = np.random.choice(178, 128, replace=False)

test\_idx = np.array(list(set(range(178)) - set(train\_idx)))

X\_train, Y\_train = X[train\_idx], Y[train\_idx]

X\_test, Y\_test = X[test\_idx], Y[test\_idx]

class KnnNet(nn.Cell):

    def \_\_init\_\_(self, k):

        super(KnnNet, self).\_\_init\_\_()

        self.tile = P.Tile()

        self.sum = P.ReduceSum()

        self.topk = P.TopK()

        self.k = k

    def construct(self, x, X\_train):

        # Tile input x to match the number of samples in X\_train

        x\_tile = self.tile(x, (128, 1))

        square\_diff = F.square(x\_tile - X\_train)

        square\_dist = self.sum(square\_diff, 1)

        dist = F.sqrt(square\_dist)

        # -dist mean the bigger the value is, the nearer the samples are

        values, indices = self.topk(-dist, self.k)

        return indices

def knn(knn\_net, x, X\_train, Y\_train):

    x, X\_train = ms.Tensor(x), ms.Tensor(X\_train)

    indices = knn\_net(x, X\_train)

    topk\_cls = [0]\*len(indices.asnumpy())

    for idx in indices.asnumpy():

        topk\_cls[Y\_train[idx]] += 1

    cls = np.argmax(topk\_cls)

return cls

acc = 0

knn\_net = KnnNet(5)

for x, y in zip(X\_test, Y\_test):

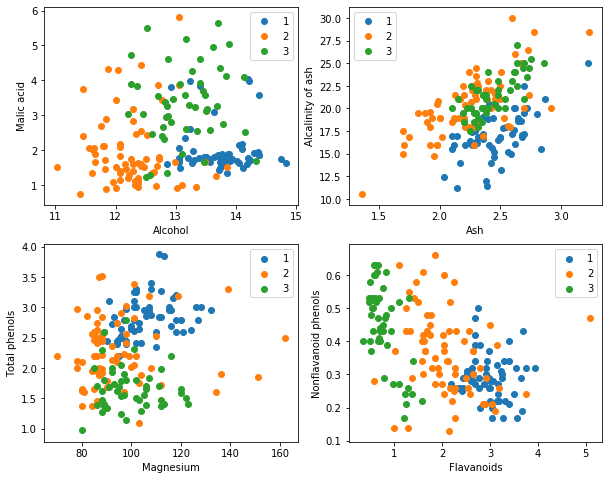
    pred = knn(knn\_net, x, X\_train, Y\_train)

    acc += (pred == y)

    print('label: %d, prediction: %s' % (y, pred))

print('Validation accuracy is %f' % (acc/len(Y\_test)))

1. 实验结果



label: 1, prediction: 1  
label: 1, prediction: 1  
label: 2, prediction: 3  
label: 3, prediction: 2  
label: 1, prediction: 3  
label: 3, prediction: 3  
label: 1, prediction: 1  
label: 1, prediction: 1  
label: 3, prediction: 3  
label: 1, prediction: 1  
label: 3, prediction: 3  
label: 1, prediction: 1  
label: 1, prediction: 1  
label: 1, prediction: 1  
label: 3, prediction: 1  
label: 3, prediction: 2  
label: 1, prediction: 1  
label: 3, prediction: 2  
label: 3, prediction: 3  
label: 3, prediction: 1  
label: 3, prediction: 3  
label: 3, prediction: 3  
label: 3, prediction: 2  
label: 1, prediction: 1  
label: 1, prediction: 1  
label: 3, prediction: 2  
label: 3, prediction: 3  
label: 1, prediction: 1  
label: 3, prediction: 3  
label: 1, prediction: 1  
label: 1, prediction: 1  
label: 1, prediction: 1  
label: 2, prediction: 3  
label: 2, prediction: 1  
label: 2, prediction: 3  
label: 2, prediction: 1  
label: 2, prediction: 2  
label: 2, prediction: 3  
label: 2, prediction: 1  
label: 2, prediction: 3  
label: 2, prediction: 3  
label: 2, prediction: 2  
label: 2, prediction: 2  
label: 2, prediction: 2  
label: 2, prediction: 2  
label: 2, prediction: 3  
label: 2, prediction: 2  
label: 2, prediction: 3  
label: 2, prediction: 2  
label: 2, prediction: 2  
Validation accuracy is 0.620000

1. 实验总结
2. 了解了KNN的基本概念，并且使用MindSpore进行KNN实验
3. 掌握华为云的使用