# 1.数据的转换

# 以*txt*文件格式转换*csv*文件格式为例，转换代码如下：

import csv

def txtToCsv():

csvFile = open("Iris .csv", 'w', newline='', encoding='utf-8')

writer = csv.writer(csvFile)

csvRow = []# 用来存储csv文件中一行的数据

header = ['sepal length','sepal width','petal length','petal width','class']

writer.writerow(header)

f = open("Iris.txt", 'r', encoding='GB2312')

for line in f:

csvRow = line.split()

writer.writerow(csvRow) #将csvRow中数据写入csv文件中

f.close()

csvFile.close()

if \_\_name\_\_ == "\_\_main\_\_":

#将txt文件转换为csv格式文件

txtToCsv()

# 通过以上代码，实现将*txt*格式数据集写到*csv*格式文件中，从而实现格式转换。

# *2.*数据的读取

# 以读取*csv*文件为例，以下代码实现读取*csv*格式文件，生成数据矩阵

import numpy as np

import pandas as pd

def csv2Arr(fileName):

*"""*

准备数据*:* 从*csv*文件中解析数据

***:param*** *fileName:*数据集名称（带相对路径）

***:return****:data\_arr*：数据集数组*(*含标签，*2*维数组，类型：*ndarray)*

*"""*

df = pd.read\_csv(fileName) # 读文件返回DataFrame对象

df = pd.DataFrame(df)

data\_arr = np.array(df) # 将DataFrame转换为ndarray

return data\_arr

def operData(fileName):

*"""*

操作数据：

***:param*** *fileName:*数据集名称（带相对路径）

***:return****:*

*"""*

#读取csv格式数据，将数据转换为数据集数组

data\_arr = csv2Arr(fileName)

p = 2

q = 10

#输出全部点

print(data\_arr)

#输出第p、q个点

print(data\_arr[p-1],data\_arr[q-1])#下标从0开始

if \_\_name\_\_ == "\_\_main\_\_":

#读取数据

operData(‘Iris.csv')

# 代码中两次出现*print*输出语句。

# 第一次输出全部点（测试数据集共*30*个点）

# *[[5.1 3.5 1.4 0.2 1. ]*

# *[4.9 3. 1.4 0.2 1. ]*

# *[4.7 3.2 1.3 0.2 1. ]*

# *[4.6 3.1 1.5 0.2 1. ]*

# *[5. 3.6 1.4 0.2 1. ]*

# *[5.4 3.9 1.7 0.4 1. ]*

# *[4.6 3.4 1.4 0.3 1. ]*

# *[5. 3.4 1.5 0.2 1. ]*

# *[4.4 2.9 1.4 0.2 1. ]*

# *[4.9 3.1 1.5 0.1 1. ]*

# *[7. 3.2 4.7 1.4 2. ]*

# *[6.4 3.2 4.5 1.5 2. ]*

# *[6.9 3.1 4.9 1.5 2. ]*

# *[5.5 2.3 4. 1.3 2. ]*

# *[6.5 2.8 4.6 1.5 2. ]*

# *[5.7 2.8 4.5 1.3 2. ]*

# *[6.3 3.3 4.7 1.6 2. ]*

# *[4.9 2.4 3.3 1. 2. ]*

# *[6.6 2.9 4.6 1.3 2. ]*

# *[5.2 2.7 3.9 1.4 2. ]*

# *[6.3 3.3 6. 2.5 3. ]*

# *[5.8 2.7 5.1 1.9 3. ]*

# *[7.1 3. 5.9 2.1 3. ]*

# *[6.3 2.9 5.6 1.8 3. ]*

# *[6.5 3. 5.8 2.2 3. ]*

# *[7.6 3. 6.6 2.1 3. ]*

# *[4.9 2.5 4.5 1.7 3. ]*

# *[7.3 2.9 6.3 1.8 3. ]*

# *[6.7 2.5 5.8 1.8 3. ]*

# *[7.2 3.6 6.1 2.5 3. ]]*

# 第二次*print*输出指定点，代码中指定的是第*2*和第*10*个点。输出结果如下：

# *[4.9 3. 1.4 0.2 1. ] [4.9 3.1 1.5 0.1 1. ]*

# *3.*数据归一化处理

# 若每个特征值同等重要，某一维度的一些差距极端的数值严重影响计算结果，所以需要归一化特征值。

#归一化特征值

def autoNorm(data\_arr):

*"""*

归一化特征值：（将任意取值范围的特征值转化为*0*到*1*区间内的值*)*

公式：*newValue=*（*oldValue-min*）*/(max-min)*）

***:param*** *data\_arr:* 数据集数组*(*不含标签列*)*

***:return****: normDataArr:* 归一化后的数据集数组（不含标签列）

*"""*

# 获取数据集中每一列的最小数值(返回的是一个一维数组)

minVals = data\_arr.min(0)

# 获取数据集中每一列的最大数值(返回的是一个一维数组)

maxVals = data\_arr.max(0) #print(type(maxVals))#返回的就是一个ndarray

# 最大值与最小的差值

ranges = maxVals - minVals

# 创建一个与dataSet同shape的全0矩阵，用于存放归一化后的数据

normDataArr = np.zeros(np.shape(data\_arr))

# 得到行数矩阵行数

m = data\_arr.shape[0]

# 把最小值扩充为与dataSet同shape，然后作差即（oldValue-min）

normDataArr = data\_arr - np.tile(minVals, (m,1))

#得到归一化数据集数组 （oldValue-min）/(max-min）是指对应元素进行除法运算，而不是矩阵除法（矩阵除法在numpy中要用linalg.solve(A,B)）

normDataArr = normDataArr/np.tile(ranges, (m,1))

return normDataArr

# 以上为归一化具体实现代码。下面是调用及输出代码

# 归一化特征值（参数不含标签列）

normDataArr = autoNorm(data\_arr[:,0:-1])

# 特征值拼接标签列（得到标准化后的含标签的数组集）

data\_arr =np.c\_[data\_arr,data\_arr[:,-1]]

print(data\_arr)

# *print*输出语句最终打印归一化后的数据。输出结果如下：

# *[[0.21875 0.75 0.01886792 0.04166667 0.04166667]*

# *[0.15625 0.4375 0.01886792 0.04166667 0.04166667]*

# *[0.09375 0.5625 0. 0.04166667 0.04166667]*

# *[0.0625 0.5 0.03773585 0.04166667 0.04166667]*

# *[0.1875 0.8125 0.01886792 0.04166667 0.04166667]*

# *[0.3125 1. 0.0754717 0.125 0.125 ]*

# *[0.0625 0.6875 0.01886792 0.08333333 0.08333333]*

# *[0.1875 0.6875 0.03773585 0.04166667 0.04166667]*

# *[0. 0.375 0.01886792 0.04166667 0.04166667]*

# *[0.15625 0.5 0.03773585 0. 0. ]*

# *[0.8125 0.5625 0.64150943 0.54166667 0.54166667]*

# *[0.625 0.5625 0.60377358 0.58333333 0.58333333]*

# *[0.78125 0.5 0.67924528 0.58333333 0.58333333]*

# *[0.34375 0. 0.50943396 0.5 0.5 ]*

# *[0.65625 0.3125 0.62264151 0.58333333 0.58333333]*

# *[0.40625 0.3125 0.60377358 0.5 0.5 ]*

# *[0.59375 0.625 0.64150943 0.625 0.625 ]*

# *[0.15625 0.0625 0.37735849 0.375 0.375 ]*

# *[0.6875 0.375 0.62264151 0.5 0.5 ]*

# *[0.25 0.25 0.49056604 0.54166667 0.54166667]*

# *[0.59375 0.625 0.88679245 1. 1. ]*

# *[0.4375 0.25 0.71698113 0.75 0.75 ]*

# *[0.84375 0.4375 0.86792453 0.83333333 0.83333333]*

# *[0.59375 0.375 0.81132075 0.70833333 0.70833333]*

# *[0.65625 0.4375 0.8490566 0.875 0.875 ]*

# *[1. 0.4375 1. 0.83333333 0.83333333]*

# *[0.15625 0.125 0.60377358 0.66666667 0.66666667]*

# *[0.90625 0.375 0.94339623 0.70833333 0.70833333]*

# *[0.71875 0.125 0.8490566 0.70833333 0.70833333]*

# *[0.875 0.8125 0.90566038 1. 1. ]]*

# *4.*库函数的应用

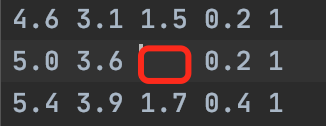
# *python*提供了许多供大家调用的库函数，当然，我们可以自己写函数。如前边写的从*csv*文件中读取数据生成矩阵的函数、归一化数据的函数，都可以加入自定义的工具包，可提高代码可读性和重用率。我们之前用到的*python*库包括*numpy*、*pandas*、*csv*。

# *numpy*库从字面上可看出主要是对数值进行操作，例如之前代码中用到的*data\_arr = np.array(df)*，就是利用*numpy*中的*array*函数，将将*DataFrame*数据类型转换为*ndarray*数据类型。

# *pandas*库是一个强大的分析结构化数据的工具集；它的使用基础是*Numpy*（提供高性能的矩阵运算）；用于数据挖掘和数据分析，同时也提供数据清洗功能。例如之前代码中用到*df = pd.read\_csv(fileName)* ，就是用到*pandas*模块中的*read\_csv*函数，用于读*csv*文件返回*DataFrame*对象。

# *csv*库以*csv*格式读取和写入表格数据。在最开始数据转换的代码中我们用到*writer = csv.writer(csvFile)*，就是使用了*csv*库中的*writer*函数，得到一个*writer*对象，用于后续像*csv*文件中写数据。

# *5.*空缺值的填充

**

在数据预处理时，若发现有缺失数据的情况，如图中红框处。一般可采取用该条数据的前一条与后一条数据在该维度的数的平均值来填充。上例子中应填充1.6。