**实验五** kNN算法原理及实现

**一** [**实验介绍**](http://demo.cjudge.net:8081/admin/courseAdmin/expAdmin/guides/editExpGuide.jsp?guideID=85#collapseOne593)

**1. 实验内容**

    本实验包括对kNN算法原理的介绍，kNN算法的步骤流程，以及如何自己动手实现kNN算法。

**2. 实验目标**

    通过本实验掌握kNN算法的原理，熟悉kNN算法。

**3. 实验知识点**

kNN算法原理；kNN算法流程

**4. 实验环境**

python 3.6.5

**5. 预备知识**

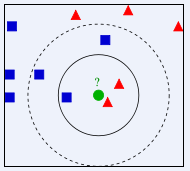
初等数学知识；Python编程基础

**二**[**实验原理**](http://demo.cjudge.net:8081/admin/courseAdmin/expAdmin/guides/editExpGuide.jsp?guideID=85#collapseOne594)

**1. kNN算法简介**

    k近邻法(k-nearest neighbor, kNN)是1967年由Cover T和Hart P提出的一种基本分类与回归方法。它的工作原理是：存在一个样本数据集合，也称作为训练样本集，并且样本集中每个数据都存在标签，即我们知道样本集中每一个数据与所属分类的对应关系。输入没有标签的新数据后，将新的数据的每个特征与样本集中数据对应的特征进行比较，然后算法提取样本最相似数据(最近邻)的分类标签。一般来说，我们只选择样本数据集中前k个最相似的数据，这就是k-近邻算法中k的出处，通常k是不大于20的整数。最后，选择k个最相似数据中出现次数最多的分类，作为新数据的分类。

    所谓K最近邻，就是k个最近的邻居的意思，说的是每个样本都可以用它最接近的k个邻居来代表。kNN算法的核心思想是如果一个样本在特征空间中的k个最相邻的样本中的大多数属于某一个类别，则该样本也属于这个类别，并具有这个类别上样本的特性。该方法在确定分类决策上只依据最邻近的一个或者几个样本的类别来决定待分样本所属的类别。 kNN方法在类别决策时，只与极少量的相邻样本有关。由于kNN方法主要靠周围有限的邻近的样本，而不是靠判别类域的方法来确定所属类别的，因此对于类域的交叉或重叠较多的待分样本集来说，kNN方法较其他方法更为适合。



    上图中，绿色圆要被决定赋予哪个类，是红色三角形还是蓝色四方形？如果K=3，由于红色三角形所占比例为2/3，绿色圆将被赋予红色三角形那个类，如果K=5，由于蓝色四方形比例为3/5，因此绿色圆被赋予蓝色四方形类。

    K最近邻(k-Nearest Neighbor，KNN)分类算法，是一个理论上比较成熟的方法，也是最简单的机器学习算法之一。该方法的思路是：如果一个样本在特征空间中的k个最相似(即特征空间中最邻近)的样本中的大多数属于某一个类别，则该样本也属于这个类别。KNN算法中，所选择的邻居都是已经正确分类的对象。该方法在定类决策上只依据最邻近的一个或者几个样本的类别来决定待分样本所属的类别。 KNN方法虽然从原理上也依赖于极限定理，但在类别决策时，只与极少量的相邻样本有关。由于KNN方法主要靠周围有限的邻近的样本，而不是靠判别类域的方法来确定所属类别的，因此对于类域的交叉或重叠较多的待分样本集来说，KNN方法较其他方法更为适合。  
    KNN算法不仅可以用于分类，还可以用于回归。通过找出一个样本的k个最近邻居，将这些邻居的属性的平均值赋给该样本，就可以得到该样本的属性。更有用的方法是将不同距离的邻居对该样本产生的影响给予不同的权值(weight)，如权值与距离成反比。

2. kNN算法流程

1. 计算已知类别数据集中的点与当前点之间的距离；
2. 按照距离递增次序排序；
3. 选取与当前点距离最小的k个点；
4. 确定前k个点所在类别的出现频率；
5. 返回前k个点所出现频率最高的类别作为当前点的预测分类。

**三**[**基础知识回顾**](http://demo.cjudge.net:8081/admin/courseAdmin/expAdmin/guides/editExpGuide.jsp?guideID=85#collapseOne596)

本实验中会涉及到Python编程的基础知识，如果你Python编程知识已经忘得差不多了，请务必仔细完成本小节的实验。Python基础编程知识回归包括文件和目录的基本操作知识，列表的基本操作，字典的基本操作，字符串的基本操作，以及numpy库的基本操作。

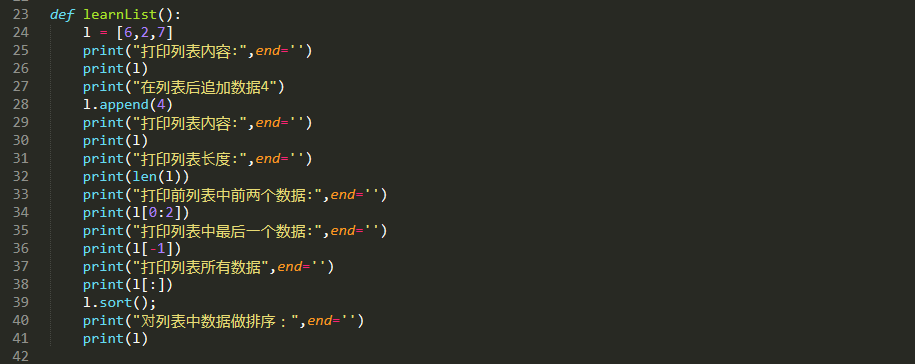
**1. Python文件和目录**

****

文件和目录的Python代码如上图所示，旨在帮你回忆起Python文件和目录的基本知识。包括：

* 如何打开文件
* 如何按行遍历文件
* 如何读取一行
* 如何遍历目录中所有文件

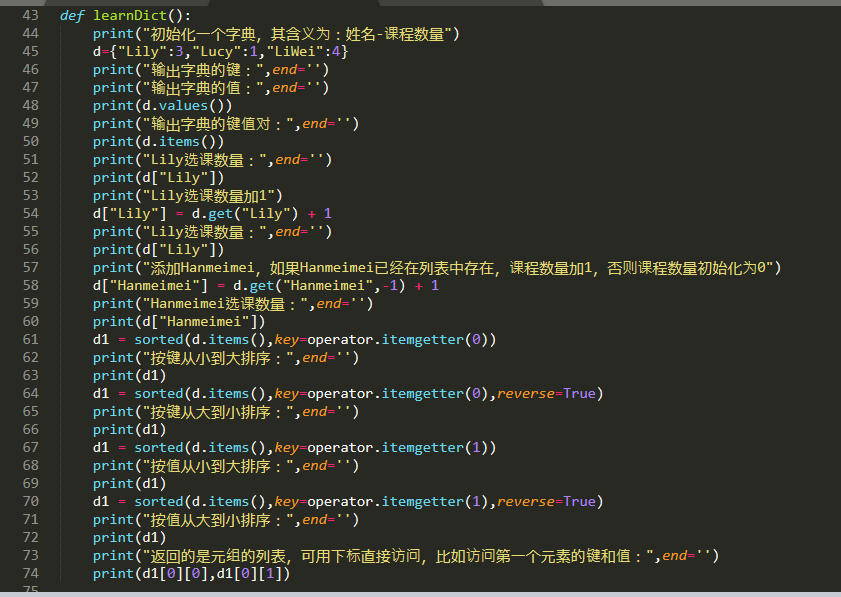
**2. Python列表**

****

列表操作的Python代码如上图所示，旨在帮助你回忆起Python列表操作的基本知识。包括：

* 如何定义列表变量
* 如何往列表中追加数据
* 如何获取列表元素数量
* 如何打印列表内容
* 如何使用：访问列表指定范围内的数据
* 如何访问列表最后一个数据
* 如何对列表做排序

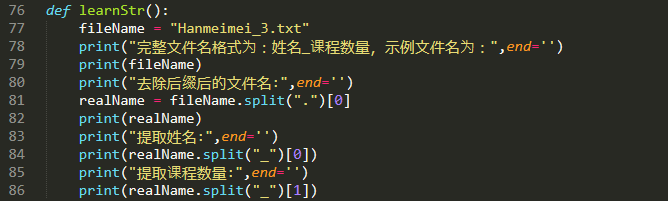
**3. Python字典**

****

字典操作的代码如上图所示，旨在帮助回忆起Python字典操作的基本知识。包括：

* 如何定义字典变量
* 如何获取字典的键集合、值集合、键值对集合
* 如何往字典中插入新的键值对（支持默认值的方式）
* 如何修改字典中的值
* 如何对字典按照键排序（正序，逆序）
* 如何对字典按照值排序（正序，逆序）
* 如何访问字典排序后的结果

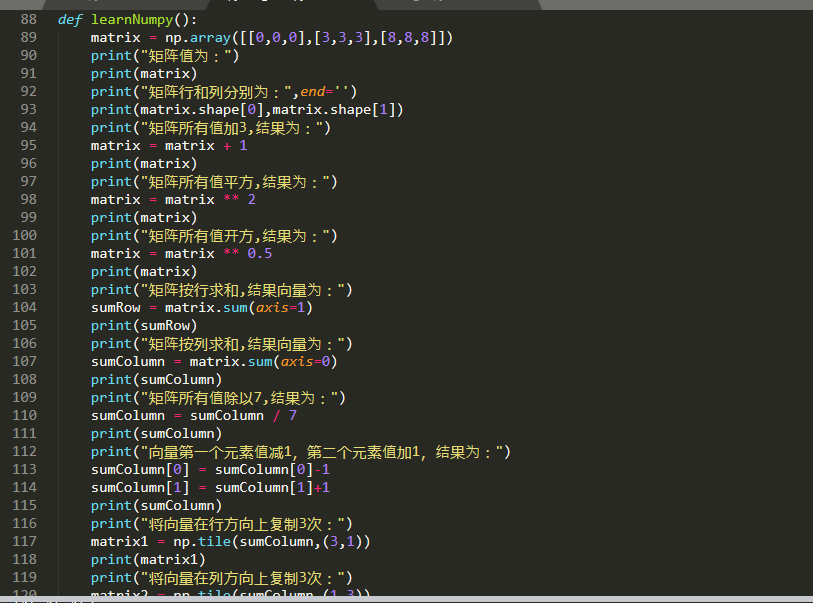
**4. Python字符串操作**

****

字典操作的代码如上图所示，旨在帮助回忆起Python字符串操作的基本知识。包括：

* 如何分割字符串
* 如何访问分割后的字符串数组

**5. Python的numpy库的基本使用**

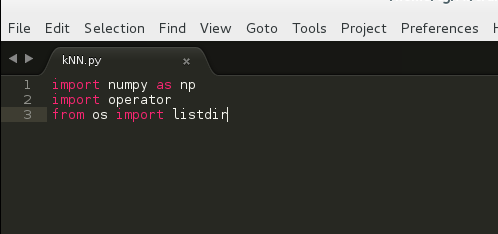
****

字典操作的代码如上图所示，旨在帮助回忆起Python字典操作的基本知识。包括：

* 如何初始化矩阵
* 如何对矩阵的每个元素值做加、减、乘、除、乘方、开方等基本运算
* 如何对矩阵按行或者按列求和
* 如何访问矩阵或向量中指定元素的值
* 如何基于向量使用tile函数构造矩阵
* 如何做矩阵间的运算。

**5**[**实验步骤：引言**](http://demo.cjudge.net:8081/admin/courseAdmin/expAdmin/guides/editExpGuide.jsp?guideID=85#collapseOne597)

接下来的实验步骤将引导你基于Python一步一步编写出自己的kNN算法。首先，在工作目录下创建kNN.py文件，第一个版本的代码将全部放在这个文件里。在kNN.py文件的顶部添加如下三行代码，如下图所示，引入我们系统需要用到的Python的相关库。其中，numpy为Python的数值计算扩展库，operator为操作符扩展库，listdir提供了可以查看目录文件列表的函数。



**6**[**实验步骤：实现kNN核心算法**](http://demo.cjudge.net:8081/admin/courseAdmin/expAdmin/guides/editExpGuide.jsp?guideID=85#collapseOne598)

如前所述，kNN算法流程如下：

1. 计算已知类别数据集中的点与当前点之间的距离；
2. 按照距离递增次序排序；
3. 选取与当前点距离最小的k个点；
4. 确定前k个点所在类别的出现频率；
5. 返回前k个点所出现频率最高的类别作为当前点的预测分类。

在kNN.py中，添加一个函数classify0作为kNN算法的核心函数，该函数的完整形式为：

def classify0(inX, dataSet, labels, k):

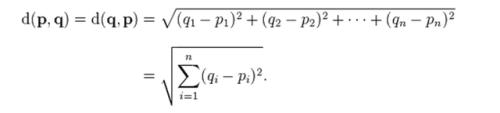
 其中各个参数的含义如下：

* inX - 用于要进行分类判别的数据(来自测试集)
* dataSet - 用于训练的数据(训练集)
* labes - 分类标签
* k - kNN算法参数,选择距离最小的k个点

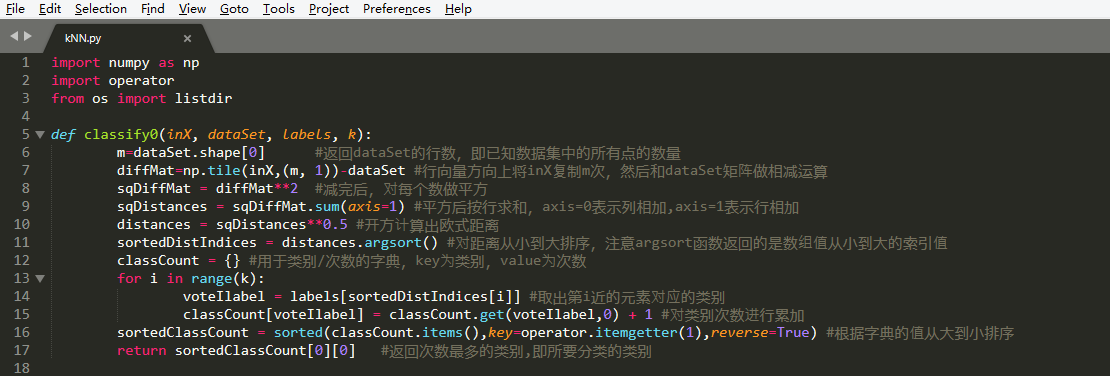
在上述参数列表中，dataSet为所有训练数据的集合，也就是表示所有已知类别数据集中的所有点，dataSet为一个矩阵，其中每一行表示已知类别数据集中的一个点。inX为一个向量，表示当前要判别分类的点。按照上述算法流程，我们首先应该计算inX这个要判别分类的点到dataSet中每个点之间的距离。dataSet中每个点也是用一个向量表示的，点与点之间的距离怎么计算呢？没错，就是求两向量之间的距离，数学上，我们知道有很多距离计算公式，包括但不限于：

* 欧氏距离
* 曼哈顿距离
* 切比雪夫距离
* 闵可夫斯基距离
* 标准化欧氏距离
* 马氏距离
* 夹角余弦
* 汉明距离
* 杰卡德距离& 杰卡德相似系数
* 信息熵

这里，我们选择最简单的欧式距离计算方法。设p和q为两向量，则两向量间的欧氏距离为：



在算法流程，输入参数含义，以及距离计算公式都明确了以后，按照kNN算法的流程，我们就可以实现kNN算法了。这里，我们使用numpy提供的各种功能来实现该算法，相比于自己纯手写各种线性代数变换操作，使用numpy的效率要高的多。classify0的实现如下：



classify0函数的完整实现如上图所示，第6行获得数据集的行数，即已知数据集中所有点的数量。第7到第10行为计算要分类的点到数据集中每个点的距离，计算过程中用到很多numpy的特性。第11行按照距离远近从近到远排序，注意argsort返回的是原始下标排序后的下标序列。从第12行到底15，统计距离最近的前k个点中每个类别出现的次数。第16行到第17行取出出现频率最高的分类。

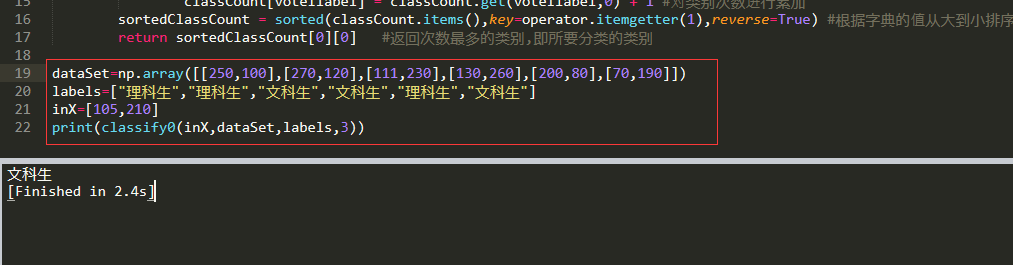
至此，kNN的算法就已经写完了。

**7**[**实验步骤：kNN算法实验**](http://demo.cjudge.net:8081/admin/courseAdmin/expAdmin/guides/editExpGuide.jsp?guideID=85#collapseOne599)

接下来，我们用个小例子验证一下kNN算法，随机挑选的6位高中生，分别让他们做文科综合试卷的分数和理科综合试卷的分数，下表为分数以及分类信息。

|  |  |  |  |
| --- | --- | --- | --- |
| **编号** | **理综成绩** | **文综成绩** | **类别** |
| 1 | 250 | 100 | 理科生 |
| 2 | 270 | 120 | 理科生 |
| 3 | 111 | 230 | 文科生 |
| 4 | 130 | 260 | 文科生 |
| 5 | 200 | 80 | 理科生 |
| 6 | 70 | 190 | 文科生 |

直觉上，理科生的理综成绩比较高，文综成绩较低，文科生的文综成绩较高，理综成绩较高。基于这些信息，我们利用kNN算法判断成绩为（105,210）所属的分类，代码实现如下：



运行结果如下显示“文科生”，输出结果结果比较符合预期。

**8**[**实验总结**](http://demo.cjudge.net:8081/admin/courseAdmin/expAdmin/guides/editExpGuide.jsp?guideID=85#collapseOne600)

**1. 指导思想**

    kNN算法的指导思想是“近朱者赤，近墨者黑”，由你的邻居来推断出你的类别。先计算待分类样本与已知类别的训练样本之间的距离，找到距离与待分类样本数据最近的k个邻居；再根据这些邻居所属的类别来判断待分类样本数据的类别。用空间内两个点的距离来度量。距离越大，表示两个点越不相似。距离的选择有很多，通常用比较简单的欧式距离。

**2. 类别的判定**

    投票决定：少数服从多数，近邻中哪个类别的点最多就分为该类。

    加权投票法：根据距离的远近，对近邻的投票进行加权，距离越近则权重越大（权重为距离平方的倒数）

**3. 优点**

* 简单，易于理解，易于实现，无需估计参数，无需训练；
* 适合对稀有事件进行分类；
* 特别适合于多分类问题(multi-modal,对象具有多个类别标签)， kNN比SVM的表现要好。

**4. 缺点**

* 懒惰算法，对测试样本分类时的计算量大，内存开销大，评分慢；
* 当样本不平衡时，如一个类的样本容量很大，而其他类样本容量很小时，有可能导致当输入一个新样本时，该样本的K个邻居中大容量类的样本占多数；
* 可解释性较差，无法给出决策树那样的规则。

**5. 常见问题**

#### k值的设定

    k值选择过小，得到的近邻数过少，会降低分类精度，同时也会放大噪声数据的干扰；而如果k值选择过大，并且待分类样本属于训练集中包含数据数较少的类，那么在选择k个近邻的时候，实际上并不相似的数据亦被包含进来，造成噪声增加而导致分类效果的降低。

    如何选取恰当的K值也成为KNN的研究热点。k值通常是采用交叉检验来确定（以k=1为基准）。经验规则：k一般低于训练样本数的平方根。

#### 类别的判定方式

    投票法没有考虑近邻的距离的远近，距离更近的近邻也许更应该决定最终的分类，所以加权投票法更恰当一些。

#### 距离度量方式的选择

    高维度对距离衡量的影响：众所周知当变量数越多，欧式距离的区分能力就越差。

    变量值域对距离的影响：值域越大的变量常常会在距离计算中占据主导作用，因此应先对变量进行标准化。

#### 训练样本的参考原则

    学者们对于训练样本的选择进行研究，以达到减少计算的目的，这些算法大致可分为两类。第一类,减少训练集的大小。KNN算法存储的样本数据,这些样本数据包含了大量冗余数据,这些冗余的数据增了存储的开销和计算代价。缩小训练样本的方法有:在原有的样本中删掉一部分与分类相关不大的样本样本,将剩下的样本作为新的训练样本;或在原来的训练样本集中选取一些代表样本作为新的训练样本；或通过聚类,将聚类所产生的中心点作为新的训练样本。

    在训练集中，有些样本可能是更值得依赖的。可以给不同的样本施加不同的权重，加强依赖样本的权重，降低不可信赖样本的影响。

#### 性能问题

    kNN是一种懒惰算法，而懒惰的后果：构造模型很简单，但在对测试样本分类地的系统开销大，因为要扫描全部训练样本并计算距离。已经有一些方法提高计算的效率，例如压缩训练样本量等。

**9**[**参考文献及延伸阅读**](http://demo.cjudge.net:8081/admin/courseAdmin/expAdmin/guides/editExpGuide.jsp?guideID=85#collapseOne601)

**参考资料：**

1. 哈林顿，李锐. 机器学习实战 : Machine learning in action[M]. 人民邮电出版社, 2013.
2. 周志华. 机器学习:Machine learning[M]. 清华大学出版社, 2016.

**延伸阅读：**

1. 李航. 统计学习方法[M]. 清华大学出版社, 2012.