山东大学 软件 学院

模式识别 课程实验报告

|  |  |  |  |
| --- | --- | --- | --- |
| 学号：201500301229 | 姓名：李洪德 | | 班级：15级二班 |
| 实验题目：k-Means算法设计与实现 | | | |
| 实验学时：4 | | 实验日期：2017.12.18 | |
| 实验目的：  了解并掌握K聚类算法的原理与实现 | | | |
| 硬件环境：  Dell laptop  Intel i7 5500U | | | |
| 软件环境：  操作系统：Windows 8.1  使用语言：Python 3.6  开发软件包：Numpy  编译器：pycharm | | | |
| 实验步骤与内容：  实验数据分析：  **红酒数据集（Wine Data Set）**[**http://archive.ics.uci.edu/ml/datasets/Wine**](http://archive.ics.uci.edu/ml/datasets/Wine)  共178个数据，每个数据特征为13维  13个特征分别为：（13个化学成分，每个成分取值为实数）  1) Alcohol  2) Malic acid  3) Ash  4) Alcalinity of ash  5) Magnesium  6) Total phenols  7) Flavanoids  8) Nonflavanoid phenols  9) Proanthocyanins  10) Color intensity  11) Hue  12) OD280/OD315 of diluted wines  13) Proline  在wine.txt文件中，每份红酒数据为一行，第一列为红酒所属类别。用于分析聚类算法的效果。  算法设计：  首先写距离函数，这里我用的欧氏距离  *#计算两个向量之间的欧氏距离* **def** distEclud(vecA, vecB):  **return** sqrt(sum(power(vecA-vecB,2)))  写加载数据的函数，由于红酒数据集各项上都是数值类型，将数据读进来之后要变成浮点数：  **def** loadDataSet(filename):  dataMat = []  lineClass = []  **with** open(filename) **as** fr:  **for** line **in** fr.readlines():  curLine = line.strip().split(**','**)  fltLine = list(map(float,curLine))  dataMat.append(fltLine[1:])  lineClass.append(int(curLine[0]))  **return** mat(dataMat),lineClass  为给定数据集创建一个包含K个随机质心的集合。随机质心要在整个边界之内。通过读取每一列的最大最小值，然后生成0到1之间的随机数并通过最小值和范围确保质心在数据的边界之内。  *#为给定数据集构建一个包含k个随机质心的集合* **def** randCent(dataSet, k):  n = shape(dataSet)[1]  centroids = mat(zeros((k,n)))  **for** j **in** range(n):  minJ = min(dataSet[:,j])  rangeJ = float(max(dataSet[:,j])-minJ)  centroids[:,j] = minJ + rangeJ\*random.rand(k,1)  **return** centroids  创建女个点作为起始质心（经常是随机选择） 当任意一个点的簇分配结果发生改变时  对数据集中的每个数据点 . 对每个质心  计算质心与数据点之间的距离  将数据点分配到距其最近的簇  对每一个簇  计算簇中所有点的均值并将均值作为质心  **def** kMeans(dataSet, k, distMeas=distEclud, createCent=randCent):  m = shape(dataSet)[0]  clusterAssment = mat(zeros((m,2)))   centroids = createCent(dataSet, k)  clusterChanged = **True  while** clusterChanged:  clusterChanged = **False  for** i **in** range(m):  minDist = inf; minIndex = -1  **for** j **in** range(k):  distJI = distMeas(centroids[j,:],dataSet[i,:])  **if** distJI < minDist:  minDist = distJI; minIndex = j  **if** clusterAssment[i,0] != minIndex: clusterChanged = **True** clusterAssment[i,:] = minIndex,minDist\*\*2  print(centroids)  **for** cent **in** range(k):*#重新计算质心* ptsInClust = dataSet[nonzero(clusterAssment[:,0].A==cent)[0]]*#得到当前簇内所有点* centroids[cent,:] = mean(ptsInClust, axis=0)  **return** centroids, clusterAssment  但是这样之后，分类效果并不好，可以说是很差，查看数据集发现，最后一项数据比其他项的数值要大得多。所以这里进行归一化处理，找到每个属性上的最大值和最小值，（每一项-最小值）/（最大值-最小值）来将数据值变换到0-1之间。  *#数据预处理 #进行归一化之后分类器效果大幅提高* **def** autoNorm(dataSet):  minVals = dataSet.min(0)*#每列的最小值* maxVals = dataSet.max(0)  ranges = maxVals-minVals  m=dataSet.shape[0]  normDataSet = dataSet-tile(minVals, (m,1))  normDataSet = normDataSet/tile(ranges,(m,1))  **return** normDataSet, ranges, minVals  代码运行:  K=3时  >>> import kMeans  >>> dataMat,lineClass=kMeans.loadDataSet('wine.txt')  >>> normData,ranges,minVals=kMeans.autoNorm(dataMat)  >>> centroids,cluster=kMeans.kMeans(normData,3)    可以看到，聚类效果比较好，只有极少数的样本点被分错。  K=4:    从上面可以看到，k=3聚类效果和准确性时最好的，这是因为预先知道这个数据集时分为3类的。而当k=4时，K-均值算法收敛但聚类效果较差的原因是，K-均值算法收敛到了局部最小值，而非全局最小值  上面clusterAssment的第二列值代表了误差平方和，这个平方和越小，代表聚类效果越好。  二分-K均值算法，是一种首先分成一个簇，然后逐渐分割的方法，这样能得到误差平方和最小的划分。 | | | |
| 结论分析与体会： | | | |