Code讲解

刘春花 姜姗

PCA-伪代码

1. 去除平均值
2. 计算协方差矩阵
3. 计算协方差矩阵的特征值和特征向量
4. 将特征值从小到大排序
5. 保留最大的N个特征向量
6. 将数据转换到上述N个特征向量构建的空间中

PCA-背景知识

协方差：

$$cov(X_i, X_j) = E[(X_i - E(X_i))(X_j - E(X_j))].$$

协方差矩阵：

n 维随机变量 $X = (X_1, X_2, \cdots, X_n)^T$ 的协方差矩阵定义为

$$C := C(X) = (c_{i,j})_{n\times n} = \begin{bmatrix} cov(X_1, X_1) & cov(X_1, X_2) & \cdots & cov(X_1, X_n) \\ cov(X_2, X_1) & cov(X_2, X_2) & \cdots & cov(X_2, X_n) \\ & & \ddots & \\ cov(X_n, X_1) & cov(X_n, X_2) & \cdots & cov(X_n, X_n) \end{bmatrix}_{n\times n},$$

PCA-数据读取

```
from numpy import *

def loadDataSet(fileName, delim='\t'):
    fr = open(fileName)
    stringArr = [line.strip().split(delim) for line in fr.readlines()]
    datArr = [map(float,line) for line in stringArr]
    return mat(datArr)
```

PCA-算法部分

```
def pca(dataMat, topNfeat=9999999):
    meanVals = mean(dataMat, axis=0)   #求均值
    meanRemoved = dataMat - meanVals   # 减均值
    covMat = cov(meanRemoved, rowvar=0) #计算协方差矩阵
    eigVals,eigVects = linalg.eig(mat(covMat))   #计算协方差矩阵的特征值和特征向量
    eigValInd = argsort(eigVals)                 #对特征向量按照索引从小到大进行排序
    eigValInd = eigValInd[:-(topNfeat+1):-1]   #取topK个特征值的索引
    redEigVects = eigVects[:,eigValInd]          #根据特征值的索引找对应的特征向量，构建新空间的特征矩阵
    lowDDataMat = meanRemoved * redEigVects    #将数据映射到降维后的空间中。
    return lowDDataMat
```

kMeans-伪代码

1. 创建k个点作为起始质心(随机初始化/随机选择)
2. While:当任意一个点的簇分配结果发生改变时
    for:数据集中的每一个数据点
        for: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
3. for:每一个簇
        计算簇中的所有点的均值并将其均值作为质心

kMeans-从文件中读取数据

```
from numpy import *

def loadDataSet(fileName):
    dataMat = []
    fr = open(fileName)
    for line in fr.readlines():
        curLine = line.strip().split('\t')
        fltLine = map(float,curLine)
        dataMat.append(fltLine)
    return dataMat
```

kMeans –计算向量距离

```
def distEclud(vecA, vecB):
    return sqrt(sum(power(vecA - vecB, 2)))
```

$$d_{12} = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

$$d_{12} = \sqrt{\sum_{k=1}^{n} (x_{1k} - x_{2k})^2}$$

kMeans-随机初始化质心

```
def randCent(dataSet, k):
    n = shape(dataSet)[1]#取样本维数
    centroids = mat(zeros((k,n)))#创建质心矩阵，大小为k*n
    for j in range(n): #填充质心矩阵
        minJ = min(dataSet[:,j]) #找到数据集的第j维上的最小值
        rangeJ = float(max(dataSet[:,j]) - minJ)#同理找到最大值，相减得到取值范围
        centroids[:,j] = mat(minJ + rangeJ * random.rand(k,1))#生成随机数，使之随机点在数据边界内
    return centroids
```

kMeans-算法部分

```
def kMeans(dataSet, k, distMeas=distEclud, createCent=randCent):
    m = shape(dataSet)[0]       #计算所有样本总数
    clusterAssment = mat(zeros((m,2)))# 存放簇信息和距离信息

    centroids = createCent(dataSet, k)# 随机创建k个质心
    clusterChanged = True               #控制while循环的停止/继续
    while clusterChanged:
        clusterChanged = False
        for i in range(m):...
        print centroids
        for cent in range(k):#重新计算质心
            ptsInClust = dataSet[nonzero(clusterAssment[:,0].A==cent)[0]]#获得当前簇的所有样本
            centroids[cent,:] = mean(ptsInClust, axis=0) #利用当前簇所有样本计算当前簇的质心
    return centroids, clusterAssment
```

代码网址

- https://github.com/blcunlp/ML-DMcourse/tree/master/Course%20One

Your Coding Time!!!

Thank You ！ ！ ！