首都师范大学

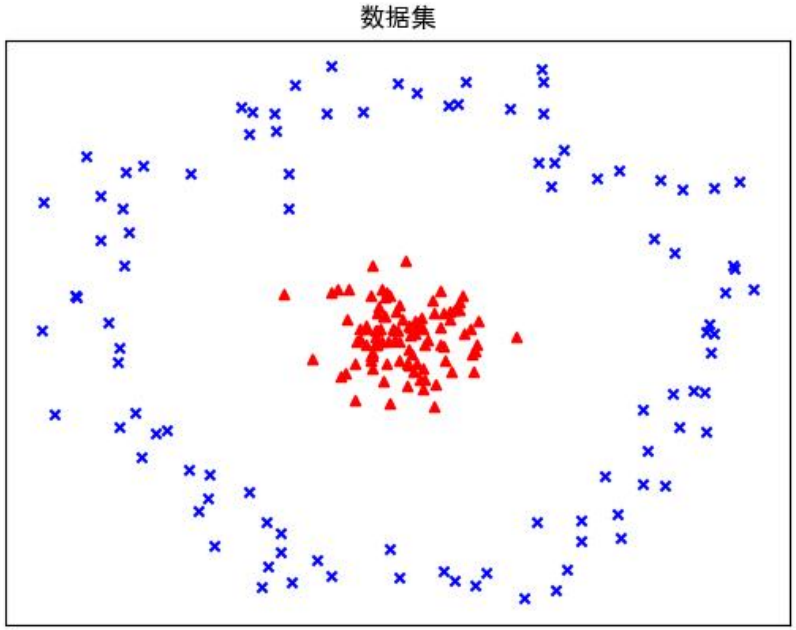
人工智能数学基础Python实践
第十一章 熵与激活函数

张苗苗

信息工程学院

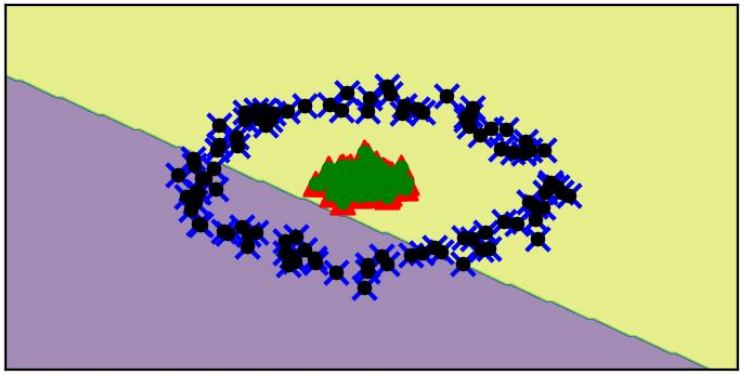
课程回顾-- 非线性SVM实现

随机生成如图所示的两类数据，构造三种核函数（线性、多项式、高斯）的算法拟合数据集，并画出拟合的分类超平面（P306，10.7.2）

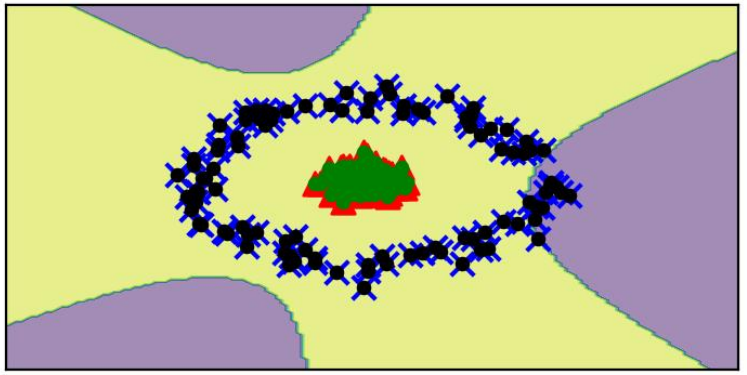


课程回顾-- 非线性SVM实现

线性核函数



多项式核函数



$$K(x, y) = x \cdot y$$

$$K(x, y) = [\gamma(x \cdot y) + c]^d$$

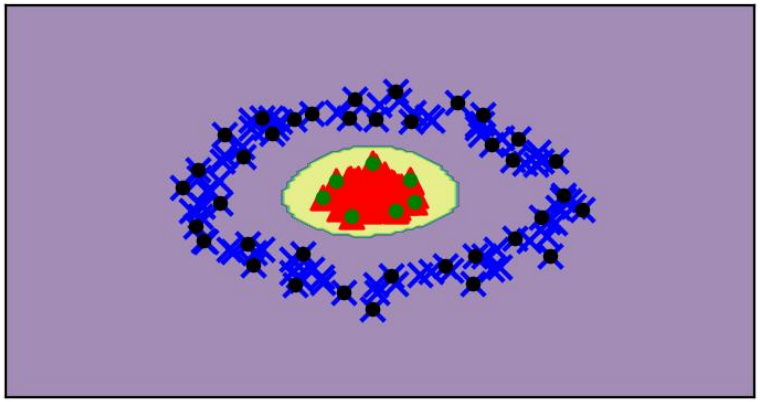
#实例化，设置核函数

```
model_linear = SVC(C=1.0, kernel='linear')
model_poly = SVC(C=1.0, kernel='poly', degree=3,gamma="auto")
```

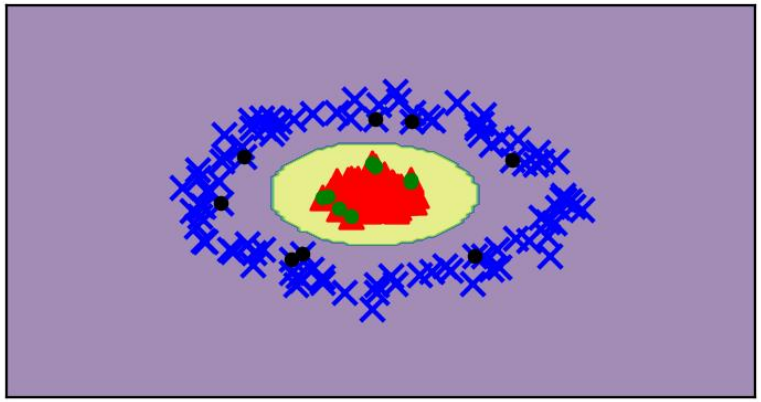
课程回顾-- 非线性SVM实现

$$K(x,y) = e^{(-\frac{\|x-y\|^2}{2\sigma^2})}$$

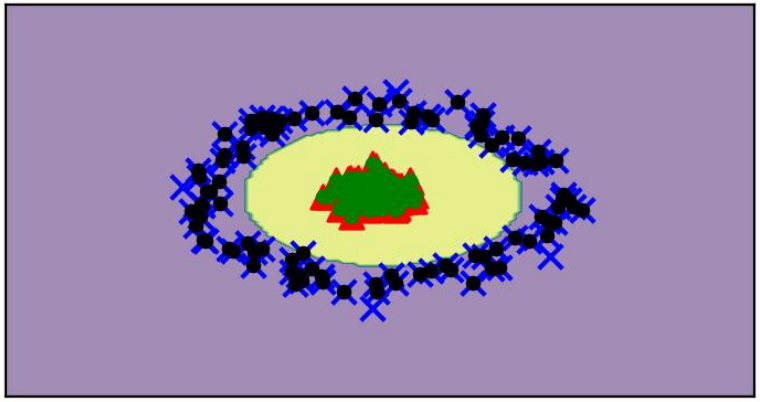
rbf函数，参数gamma=10



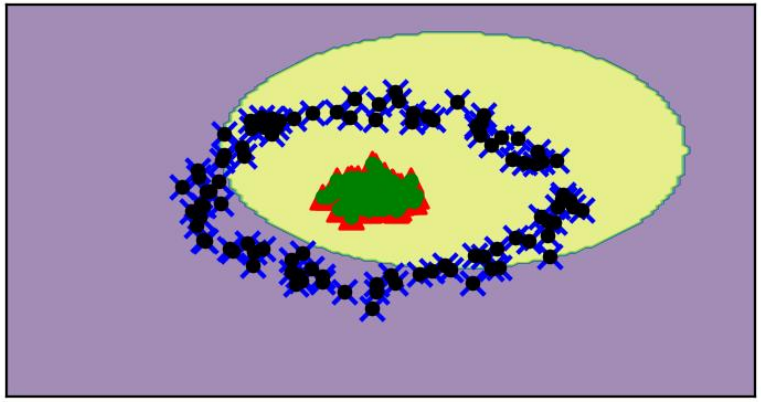
rbf函数，参数gamma=1



rbf函数，参数gamma=0.1



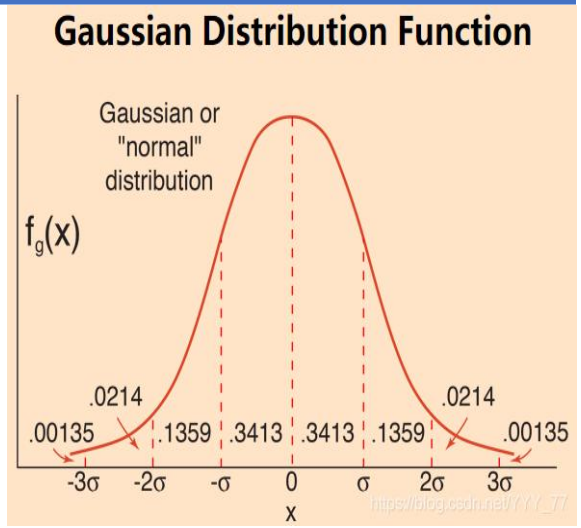
rbf函数，参数gamma=0.01



model_rtf= SVC(C=1.0, kernel='rbf', gamma=gamma)

课程回顾-- 非线性SVM实现

$$K(x,y) = e^{(-\frac{\|x-y\|^2}{2\sigma^2})}$$

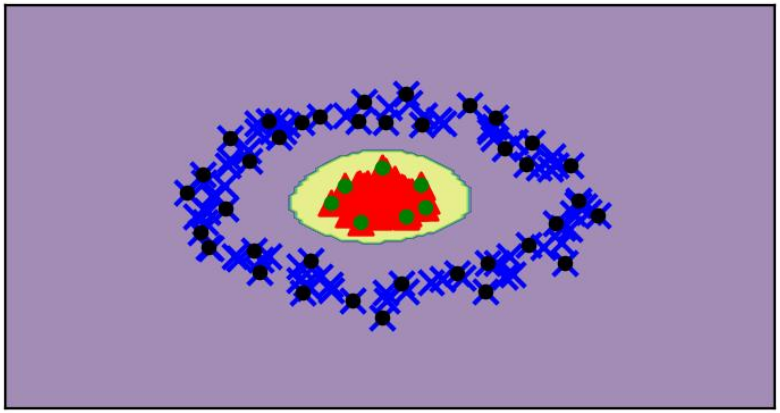


各种核函数与它们的计算公式

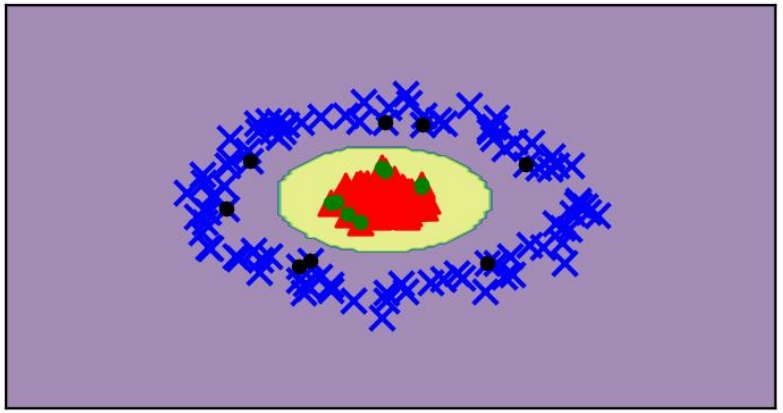
核函数	计算公式
线性核	$K(x_i,x_j)=x_i^T x_j$
多项式核	$K(x_i,x_j)=(\gamma x_i^T x_j+b)^d$
径向基函数核/高斯核	$K(x_i,x_j)=\exp(-\gamma\|x_i-x_j\|^2)$
sigmoid 核	$K(x_i,x_j)=\tanh(\gamma x_i^T x_j+b)$

课程回顾-- 非线性SVM实现

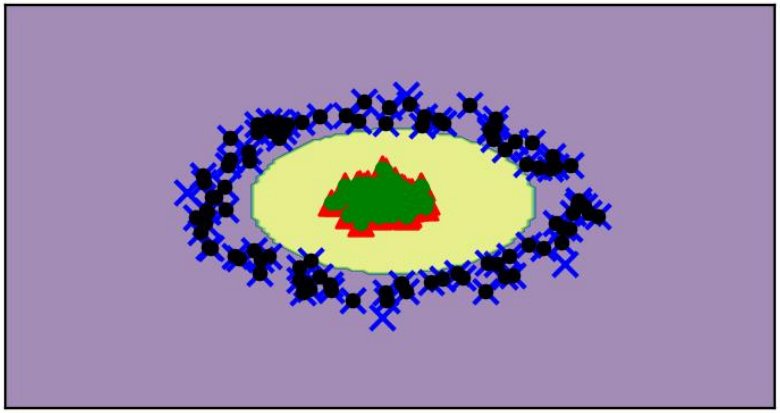
rbf函数，参数gamma=10



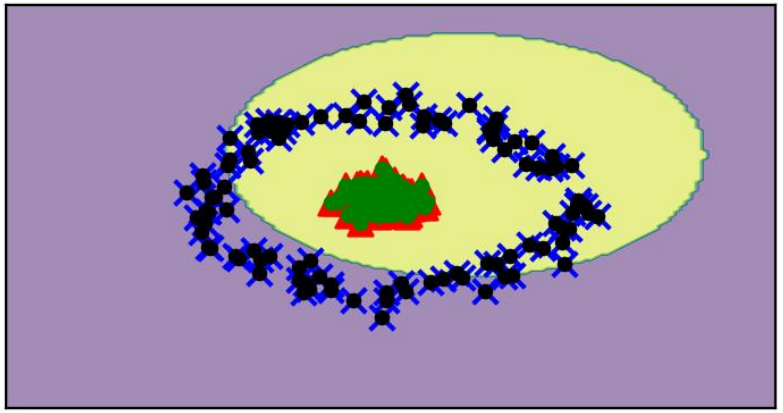
rbf函数，参数gamma=1



rbf函数，参数gamma=0.1



rbf函数，参数gamma=0.01



gamma越大，σ越小，越容易过拟合

主要内容

- 在Python中实现以下操作：
  - 熵和信息熵
  - 激活函数
  - ID3分类算法

- 所需的python库：
  - numpy
  - matplotlib
  - math

练习1-- 熵和信息熵的python实现

在python中实现不同均匀分布概率下信息熵与概率的关系，并画图显示。

均匀分布概率分别为[0.01,0.02，...，0.99]

对应概率单个事件的不确定性计算：

$$f(p_i) = -log2(p_i)$$

对应概率的信息熵计算：

$$H = E\ (f(p_i)) = -\sum_{i=1}^{n} p_i log2(p_i)$$

单个事件的信息熵分量计算：

$$f(p_i) = -p_i log2(p_i)$$

练习1-- 熵和信息熵的python实现

```
import numpy as np
import matplotlib.pyplot as plt

#生成图
plt.figure(figsize=(8,5), dpi=80)

#生成不同均匀分布的概率
X = np.linspace(0,1, 101,endpoint=True)
X=X[1:100]

#计算信息熵并画图显示
Y=-1*np.log2(X)
plt.plot(X,Y,color='red',linestyle='dashed',label='log2(p)')

#计算信息熵分量并画图显示
Y=-1*X*np.log2(X)
plt.plot(X,Y,color='green',linewidth=2,label='plog2(p)')
plt.legend()
plt.show()
```

练习2-- 常见激活函数的python实现

在python中实现Sigmoid函数及其导数，并画图显示。

Sigmoid函数：

$$f(x) = \frac{1}{1 + e^{-x}}$$

Sigmoid函数导数：

$$f'(x) = f(x)[1 - f(x)]$$

练习2-- 常见激活函数的python实现

```
import numpy as np
import matplotlib.pyplot as plt

#定义sigmoid函数及其导数
def sigmoid(x):
    s = 1 / (1 + np.exp(-x))
    return s

def sigmoid_derivative(x):
    s=sigmoid(x)
    ds = s*(1-s)
    return ds
```

练习2-- 常见激活函数的python实现

```
#生成图
plt.figure(figsize=(8,5), dpi=80)

#给定X的显示范围
X = np.linspace(-10,10, 201,endpoint=True)

#计算f(x)和其导数
Y=sigmoid(X)
dY=sigmoid_derivative(X)

#画图显示
plt.plot(X,Y,color='green',linewidth=2,label='sigmoid function')
plt.plot(X,dY,color='red',linestyle='dashed',label='sigmoid
derivative function')
plt.legend()
plt.show()
```

练习3-- ID3分类算法python实现

P339页综合案例：

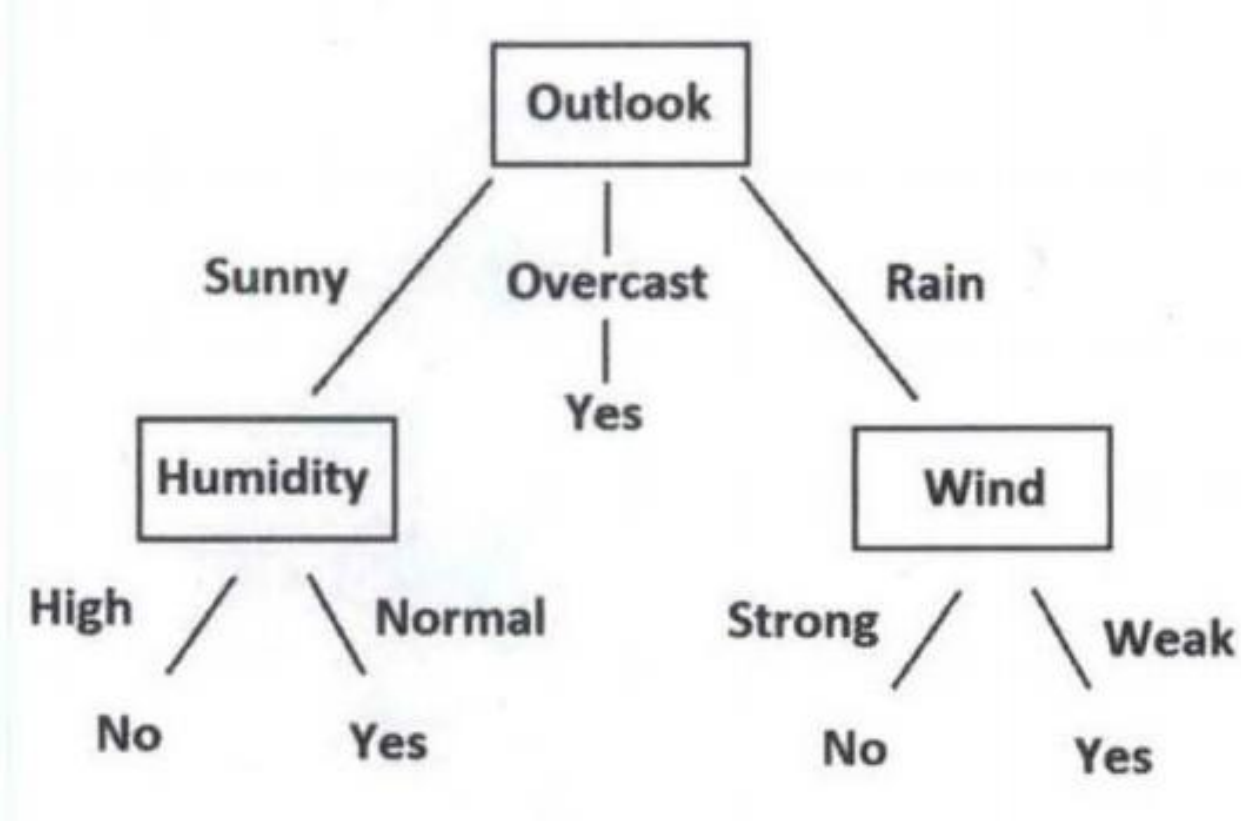
使用ID3分类算法对以下数据进行分类

表 11-1 用于 ID3 分类算法的源数据

Day	Outlook	Temp.	Humidity	Wind	Decision
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

练习3-- ID3分类算法python实现

ID3决策树分类算法的结果



练习3-- ID3分类算法python实现

```python
import math

def createDataset(): #数据集
    featurename = ['Outlook', 'Temp', 'Humidity', 'Wind', 'Decision']
    dataset = [
        ['Sunny', 'Hot', 'High', 'Weak', 'No'],
        ['Sunny', 'Hot', 'High', 'Strong', 'No'],
        ['Overcast', 'Hot', 'High', 'Weak', 'Yes'],
        ['Rain', 'Mild', 'High', 'Weak', 'Yes'],
        ['Rain', 'Cool', 'Normal', 'Weak', 'Yes'],
        ['Rain', 'Cool', 'Normal', 'Strong', 'No'],
        ['Overcast', 'Cool', 'Normal', 'Strong', 'Yes'],
        ['Sunny', 'Mild', 'High', 'Weak', 'No'],
        ['Sunny', 'Cool', 'Normal', 'Weak', 'Yes'],
        ['Rain', 'Mild', 'Normal', 'Weak', 'Yes'],
        ['Sunny', 'Mild', 'Normal', 'Strong', 'Yes'],
        ['Overcast', 'Mild', 'High', 'Strong', 'Yes'],
        ['Overcast', 'Hot', 'Normal', 'Weak', 'Yes'],
        ['Rain', 'Mild', 'High', 'Strong', 'No']
    ]
    return dataset, featurename
```

练习3-- ID3分类算法python实现

```python
#根据特征划分数据，axis表示特征，value表示特征的取值
def splitDataSet(dataset,axis,value):
    subdataset=[]
    for featVect in dataset:
        if featVect[axis]==value:
            reduce_featVect = featVect[0:axis] # 删除这一特征
            reduce_featVect.extend(featVect[axis+1:])
            subdataset.append(reduce_featVect)
    return subdataset
```

练习3-- ID3分类算法python实现

```
def calcShannonEnt(dataset): #计算信息熵
    numSamples = len(dataset)
    labelCounts = {} #统计每个类别出现的频数，存储在字典中
    for allFeatureVector in dataset: #遍历每个实例，统计标签的频次
        currentLabel = allFeatureVector[-1]  # 获取类别，每行数据的最后一列是label
        if currentLabel not in labelCounts.keys():
            labelCounts[currentLabel] = 0
        labelCounts[currentLabel] += 1
    entropy = 0.0
    for key in labelCounts:
        prob = float(labelCounts[key])/numSamples
        entropy -= prob * math.log2(prob)
    return entropy
```

练习3-- ID3分类算法python实现

```python
def calcConditionalEnt(dataset,i,featList,uniqueVals): #计算条件熵
    # dataset :数据集
    # i:  维度
    #featList:数据集特征列表
    #uniqueVals:数据集特征集合
    ce = 0.0
    for value in uniqueVals:
        subDataSet = splitDataSet(dataset,i,value)
        prob = len(subDataSet)/float(len(dataset)) #概率
        ce += prob*calcShannonEnt(subDataSet) #条件熵计算
    return ce
```

练习3-- ID3分类算法python实现

```
def calcInformationGain(dataset,baseEntropy,i):#计算信息增益
    featList  = [example[i] for example in dataset] # 第i维特征列表
    uniqueVals = set(featList) #换成集合，集合中每个元素不重复
    newEntropy = calcConditionalEnt(dataset,i,featList,uniqueVals)
    infoGain = baseEntropy-newEntropy
    return infoGain
```

练习3-- ID3分类算法python实现

```
## 算法框架
def BestFeatToGetSubdataset(dataset):
    #下边这句实现：除去最后一列类别标签列剩余的列数即为特征个数
    numFeature = len(dataset[0]) - 1  # 最后一列是分类
    baseEntropy = calcShannonEnt(dataset)  # 返回整个数据集的信息熵
    bestInfoGain = 0.0; bestFeature = -1
    for i in range(numFeature):#遍历所有维度的特征
        infoGain = calcInformationGain(dataset, baseEntropy, i)
        if (infoGain>bestInfoGain):
            bestInfoGain = infoGain
            bestFeature = i

    return bestFeature # 返回最佳特征对应的维度
```

练习3-- ID3分类算法python实现

```
#特征若已经划分完，节点下的样本还没有统一取值，则需要进行投票
def majorityCnt(classList):
    classCount={}
    for vote in classList:
        if vote not in classCount.keys():
            classCount[vote]=0
        classCount[vote]+=1
    return max(classCount)
```

练习3-- ID3分类算法python实现

```
def createTree(dataset,featurename,chooseBestFeatureToSplitFunc=BestFeatToGetSubdataset):
    classList = [example[-1] for example in dataset] #类别列表
    if classList.count(classList[0])==len(classList): #统计属于列别classeList[0]的个数
        return classList[0] # 当类别完全相同则停止继续划分
    if len(dataset[0])==1:
        return majorityCnt(classList)
    bestFeat = chooseBestFeatureToSplitFunc(dataset)
    bestFeatLabel = featurename[bestFeat]  #最佳特征
    myTree = {bestFeatLabel:{}} #map 结构
    del (featurename[bestFeat])
    featValues = [example[bestFeat] for example in dataset]
    uniqueVals = set(featValues)
    for value in uniqueVals:
        subLabels = featurename[:]
        myTree[bestFeatLabel][value] = createTree(splitDataSet(dataset,bestFeat,value),subLabels)
    return myTree

dataset, featurename = createDataset()
myTree = createTree(dataset,featurename)
print(myTree)
```

首都师范大学

谢谢！