朴素贝叶斯分类方法

GaussianNB

1.数据集

```
Iris是鸢尾花数据集，鸢尾花包括四个属性每一个都是连续变量
所以采用GaussianNB-Bayes方法处理
```

2.代码

```
import numpy as np
from math import *
import Iris
from sklearn import datasets
iris = datasets.load_iris()
dataset = iris.data
labels = iris.target
targetnames = iris.target_names
featurenames = iris.feature_names
ParameterMat, PC = Iris.CalEverageAndDeviation(dataset, labels)
print(ParameterMat)
while(1):
    X = []
    for i in range(4):
        x = input("What's the number of {}?".format(featurenames[i]))
        X.append(x)
    Prob = Iris.Probability(ParameterMat, X, PC)
    print (Prob)
    print(Iris.SelectLabel(targetnames, Prob))


def CalEverageAndDeviation(dataset, target):    #计算某标签下的属性数据集的方差均值矩阵i为属性j为标签
    dataset = np.ravel(dataset)
    Res = np.empty([3, 4 ,2], dtype=float)
    PC = [0,0,0]
    target = list(target)
    size = len(dataset)
    dataset1 = dataset[0: size: 4]  # 第一个属性的数据
    dataset2 = dataset[1: size: 4]
    dataset3 = dataset[2: size: 4]
    dataset4 = dataset[3: size: 4]
    dataset = [dataset1, dataset2, dataset3, dataset4]
    for i in range(3):
        for j in range(4):
            if i == 2:
                begin = target.index(2)
                end = len(target)
            else:
                begin = target.index(i)
                end = target.index(i + 1)
            Res[i][j][0] = np.mean(dataset[j][begin: end-1])    #留下一个作为验证
            Res[i][j][1] = np.std(dataset[j][begin: end-1], ddof=1)
            PC[i] = (end - begin)/float(len(target))
    return Res, PC

def Probability(Res, x, PC): #根据参数返回xi对应的log概率res是参数矩阵, PC = p(C) , x是数据向量, 返回概率向量
    Prob = []
    for i in range(3):
        prob = 0
        for j in range(4):
            prob += -(x[j] - Res[i][j][0])**2/2/float(Res[i][j][1]**2) - 0.5*log(2*pi) - log(Res[i][j][1]
        prob += log(PC[i])
        Prob.append(exp(prob))
    return Prob

def SelectLabel(names, Prob):
    num = max(Prob)
    return names[Prob.index(num)]
```

3. 理论解释

我们要估计的是p(c|x), 对于不同的c我们选择其中较大的

根据朴素Bayes的思想:

$$P(c|x) = \frac{P(c)}{P(x)} \prod_{i=1}^{d} P(xi|c)$$

在假定P(x)都一样的情况下比较分子大小即可
P(c)容易获得, 而P(xi|c)应依赖于下面的公式:

$$P(xi|c) = \frac{1}{\sqrt{2\pi}\sigma_{c,i}} \exp\left(-\frac{(xi-\mu_{c,i})^2}{2\sigma_{c,i}^2}\right)$$

所以只需要根据样本数据集估算σ和μ即可:

- CalEverageAndDeviation函数分别计算第i个类别第j个属性的σ和μ写入ParameterMat中, 组成3X4numpy数组, 并同时计算P(c)
- Probability函数计算分子的概率密度大小(先取log,再用exp)
- SelectLabel取概率最大的并返回标签

4. sklearn验证

```
from sklearn.naive_bayes import GaussianNB
from sklearn import datasets
import numpy as np
clf = GaussianNB()
iris = datasets.load_iris()
dataset = iris.data[0:49]
dataset = np.concatenate((dataset, iris.data[50:99]), axis=0)
dataset = np.concatenate((dataset, iris.data[100:149]), axis=0)
target = iris.target[0:49]
target = np.concatenate((target, iris.target[50:99]),axis=0)
target = np.concatenate((target, iris.target[100:149]), axis=0)
print(dataset)
print(target)
clf.fit(dataset, target)
print(clf.theta_)
print(clf.sigma_)
print(clf.predict_proba([[5.9,3.0,5.1,1.8]]))
print(clf.predict_proba([[5.7,2.8,4.1,1.3]]))
print(clf.predict_proba([[5.0,3.3,1.4,0.2]]))
```

- 结果对比

1. 自制模型

```
What's the number of sepal length (cm)?5.9
What's the number of sepal width (cm)?3.0
What's the number of petal length (cm)?5.1
What's the number of petal width (cm)?1.8
[9.261292485519334e-142, 0.005253944318079513, 0.07204691438790524]
virginica
What's the number of sepal length (cm)?5.7
What's the number of sepal width (cm)?2.8
What's the number of petal length (cm)?4.1
What's the number of petal width (cm)?1.3
[4.144599984176061e-72, 0.45135824415136533, 7.730783371144787e-05]
versicolor
What's the number of sepal length (cm)?5.0
What's the number of sepal width (cm)?3.3
What's the number of petal length (cm)?1.4
What's the number of petal width (cm)?0.2
[2.6678772935671646, 3.9115479223810816e-17, 9.466959667143605e-25]
setosa
```

2. sklearn模型

```
[[1.39087708e-143 6.37138724e-002 9.36286128e-001]]
[[2.92766408e-73 9.99854946e-01 1.45053841e-04]]
[[1.00000000e+00 6.79092588e-18 1.15737936e-25]]
```

- 结论
  在数量级和结果上是正确的

5. 总结

1. 代码上的总结
   - numpy数组初始化可以用np.empty([a,b,c], dtype = *)或者np.zero
   - 获取某元素索引，可以用np.where(dataset = vaule),但然也可以先转化为list然后用index方法
   - np.mean, np.var, np.std获取数字特征
   - np.concatenate(a,b,axis = 0)用于多维数组的连接, 如果axis = 1, 变成每一个数组元的连接, 注意:np.append(a,b)会把连接后的数组压缩成一维的
   - np.ravel可以直接将高维数组展成一维
   - numpy数组可以直接切片，有size()方法获取长度或者len()
2. 模型上的总结
   - 取对数避免下溢出
   - sklearn是很方便的,fit(dataset, label)可以直接完成模型初始化工作, clf.predict_proba, clf.predict, clf.predict_log_proba直接提供预测概率，预测值和对数概率
   - sklearn还可以直接导入数据集