朴素贝叶斯分类算法实验报告

刘群*

地球系统科学研究中心

2015 年 4 月 14 日

这次作业是让我们编写利用朴素贝叶斯算法实现基于 Iris 数据集分类的程序. 这个实验报告的思路是先对朴素贝叶斯算法做一个简单的介绍，然后对函数做详细的介绍，最后进行总结。

1 Naive Bayes 算法简介

Naive Bayes 算法是一种基于 Bayes 概率的一种分类方法，其主要思想是要最大化后验概率。主要假设所有的特征都是相互独立的，所有的特征对结论都是同等重要的。贝叶斯分类的基本公式如下：

$$p(C|X) = \frac{p(X|C)p(C)}{p(X)} \rightarrow p(C|X) \propto p(X|C)p(C)$$

其中，$X = X(x_1, x_2, \cdots, x_n), C = C(c_1, c_2, \cdots, c_L)$. 如果 $x_1, x_2, \cdots, x_n$ 是相互独立的，则有

$$p(X|C) = p(x_1, x_2, \cdots, x_n|C) = p(x_1|C)p(x_2|C)\cdots p(x_n|C)$$

最终，我们要在计算得出的后验概率 $p(c_k|X)(k = 1, 2, \cdots, L)$ 中选择一个最大值，这样 X 就属于这个概率所对应的类别。

2 NBIris 函数介绍

这个函数是用来对 Iris 的数据进行分类的，其主要思路如下：首先通过引入的 datasets 读入所需的数据。接下来就是各种概率的计算。先计算每种类别的先验概率 $p(y)$, 然后需要计算似然概率 $p(x|y)$, 由于我们没有任何其他信息，因此这里假设每种类别中所对应的四个参数 x_1, x_2, x_3, x_4 均满足正态分布，因此就需要将对应的类别的数据提取出来，计算相应的均值和标准差，因为只需要这两个参数就可以唯一确定正态分布的函数。此时即可计算得出似然概率:

$$p(y|x) \propto p(x_i|y) = \frac{1}{\sigma_y\sqrt{2\pi}} e^{-\frac{(x_i-\mu_y)^2}{2\sigma_y^2}} \quad i = 1, 2, 3, 4$$

最后就是要计算

$$\prod_{i=1}^{4} p(x_i|y)p(y)$$

由于这样计算之后，可能出现的情况是数值特别小，这样会产生很大的误差，因此在这里我们对上式取对数，即计算

$$\ln\left(\prod_{i=1}^{4} p(x_i|y)p(y)\right) = \sum_{i=1}^{n} \ln\, p(x_i|y) + \ln\, p(y)$$

即可。最后，我们返回概率中最大的值的索引号即可。

```
 import numpy as np
from sklearn import datasets

def NBIris(x_test):
    # load train data to x_train and y_train
    iris = datasets.load_iris()
    x_train = iris.data
    y_train = iris.target

    # calculate the probability of each type
```

*电子邮件: liu-q14@mails.tsinghua.edu.cn，学号: 2014211591

```
    p_y = np.array([y_train[y_train == i].shape[0] / float(x_train.shape[0]) for i in range(3)
    ])

    # calculate the average and stddev of normal distribution
    ave = np.zeros((3,4))
    sigma = np.zeros((3,4))
    for i in range(3):
        ave[i,:] = np.average(x_train[y_train==i, :], axis=0)
        sigma[i,:] = np.std(x_train[y_train==i, :], axis=0)

    x_test_matrix = np.tile(x_test, (3,1))
    # calcualte the probability of p(xi|y) with the equation of normal distribution
    p_xy = 1.0/sigma/np.sqrt(2*np.pi)*np.exp(-(x_test_matrix-ave)**2/2.0/sigma**2)
    # change the probability to log, because the value may be too small
    log_p_yx = np.sum(np.log(p_xy), axis=1)+np.log(p_y)
    # return the index of max probability, and it's also the type label
    y_test = np.argmax(log_p_yx)
    return y_test
```

3 总结

通过这个程序的编写，我首先熟悉了朴素贝叶斯分类器的主要思想，然后进一步熟悉了 Python 的相关操作，但是还是感觉对于 numpy 库中的函数掌握不够熟练，另外，就是在利用这个库的时候，尽量采取矩阵的操作，这样可以提高程序的效率。