5.朴素贝叶斯算法

# 1 实验目的

1.理解sklearn. naive\_bayes模块中三种朴素贝叶斯算法的实现原理；

2.了解如何使用sklearn. naive\_bayes模块中各个常用算法完成机器学习任务。

# 2 实验要求

本次实验后，要求同学们能：

1.掌握朴素贝叶斯算法的实现原理；

2.学会使用Scikit-learn实现常用的朴素贝叶斯算法；

3.了解sklearn.naive\_bayes模块中高斯分布、多项式分布、伯努利分布的朴素贝叶斯算法的不同特点，了解各个超参数的含义。

# 3 实验内容

## 3.1 朴素贝叶斯算法原理简介

朴素贝叶斯算法，就是基于贝叶斯公式的一种简单、朴素的算法，它朴素在假设各个特征相互独立，互不影响，这样做大大地简化了模型，让概率计算更加简单。

（1）概念

**贝叶斯公式**：

**先验概率**：P(A)、P(B)称为先验概率，一般由经验得出。在朴素贝叶斯算法中，主要关心P(A)的值，因为在计算某一样本x出现的条件下，属于yi类的概率时，分母P(B=x)相同，只有分子的P(A=yi)与P(B|A=yi)不同，而朴素贝叶斯算法只需比较哪一个P(yi|x)的后验概率最大，不需要求得具体的值，所以可以不计算P(B)。

**后验概率**：P(A|B)称为后验概率，是基于先验概率求得的反向的概率，也是朴素贝叶斯算法的要求解的目标。

**似然度**：P(B|A)称为似然度，在朴素贝叶斯算法中一般由历史数据（训练样本得到）。

（2）基本思想

朴素贝叶斯算法的基本思想就是利用似然度和先验概率估计后验概率从而进行分类。

对于给定的待分类的样本X={a1，a2，……，an}，已知的分类类别类别Y={y1，y2，……，yn}，求解该样本X出现的条件下各个类别yi出现的后验概率P(yi |X)，哪个yi对应的P(yi |X)最大，就认为X属于yi类。

（3）计算法方法

要判断后验概率P(yi |X)的大小，就需要计算先验概率和似然度。可以在训练数据上通过频率来近似表示概率。

先用各个类别的频率估算各个类别的先验概率：

然后用频率估算各类别下各个特征属性的条件概率：

由于假设特征之间相互独立，所以a1、a2同时发生的概率为a1、a2各自概率的成绩，根据这个假设可以很简单地计算出后验概率：

最终即可估算出似然度P(yi |X)的大小，选取最大的似然度对应的yi作为X的类别。

Scikit-learn库中提供了不同的朴素贝叶斯算法模型，下面介绍最常用的高斯分布朴素贝叶斯算法、伯努利分布朴素贝叶斯算法以及多项式分布朴素贝叶斯算法。

## 3.2 高斯分布的朴素贝叶斯算法

### 3.2.1 原理

高斯分布的朴素贝叶斯算法（GaussianNB）假设特征的先验概率为正态分布，即：

其中，Y = Ck表示最终结果属于第k类，μk与是算法需要从训练集中训练出的估计值：μk为在样本类别Ck中，所有Xj的平均值。为在样本类别Ck中，所有Xj的方差。

GaussianNB适用于样本特征为连续值的样本数据。

### 3.2.2 实验

（1）导入依赖并加载数据

用Scikit-learn自带的鸢尾花数据集作为实验数据，数据集一共含有150个样本（每一类各50个样本），每一行对应一朵花，列代表每朵花的四个测量数据，分别是：花瓣的长度，宽度，花萼的长度、宽度。最后一项表示鸢尾花的类别，共三类，分别是山鸢尾（0表示）、色鸢尾（1表示）、维吉尼亚鸢尾（2表示）

导入实验数据集，并划分训练集和测试集，代码如下：

from sklearn.naive\_bayes import GaussianNB

from sklearn.datasets import load\_iris

from sklearn.model\_selection import train\_test\_split

# 导入鸢尾花数据

iris\_data = load\_iris()

data = iris\_data.data

target = iris\_data.target

# 划分数据集

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(

data,

target,

test\_size=0.2,

random\_state=1

)

（2）创建算法模型实例

用sklearn.naive\_bayes模块下GaussianNB类可以创建高斯分布的朴素贝叶斯算法模型实例，其构造器重要参数及含义如下：

**priors：用于指定各个类别的先验概率，默认为None（不人为指定）。如果用一个数组型变量进行指定，则不再根据实际数据进行调整；如果不指定，则自行根据数据计算先验概率P（Y=Ck）**

**var\_smoothing：在根据数据的方差估计**时，将最大的方差乘以**var\_smoothing并**加入估计值中，从而提高估计的稳定性

这里用默认值创建一个高斯分布朴素贝叶斯算法模型实例：

model = GaussianNB()

（3）训练并测试

该算法实例提供predict\_log\_proba、predict\_proba、fit、predict、get\_params、get\_params、score等常用方法，含义不再赘述。

调用fit()方法训练模型，调用predict()方法对测试集进行预测，调用score()方法测试模型的评分，用predict\_proba()方法查看X\_test中前五个样本属于每一个类别的概率。代码如下：

# 训练

model.fit(X\_train, Y\_train)

# 测试

print("测试数据预测值：", model.predict(X\_test))

print("测试数据实际值：", Y\_test)

# 得分

print("训练集上得分为：", model.score(X\_train, Y\_train))

print("测试集上得分为：", model.score(X\_test, Y\_test))

# X\_test中前五个样本属于每一个类别的概率

print("前五条测试数据属于每个类的概率（保留三位）：\n", np.round(model.predict\_proba(X\_test[:5]), 3))

（4）查看训练后的参数

可以用p**arameter\_（参数名加下划线）的形式直接访问训练得到的以下参数：**

**class\_count\_：每一个类别中的训练样本数量。**

**class\_prior\_：每一个类别的先验概率。**

**classes\_：分类器已知的类别标签。**

**var\_：每个类别的每个特征的方差（用于估计）。**

**theta\_：每个类别中每个特征的均值（用于估计**μk**）。**

**打印出各个参，数代码如下：**

**# 查看参数**

**print("class\_count\_", model.class\_count\_)**

**print("class\_prior\_", model.class\_prior\_)**

**print("classes\_", model.classes\_)**

**print("var\_\n", model.var\_)**

**print("theta\_\n", model.theta\_)**

### **3.2.3 实验结果**

（1）高斯分布的朴素贝叶斯算法模型的测试与评价结果

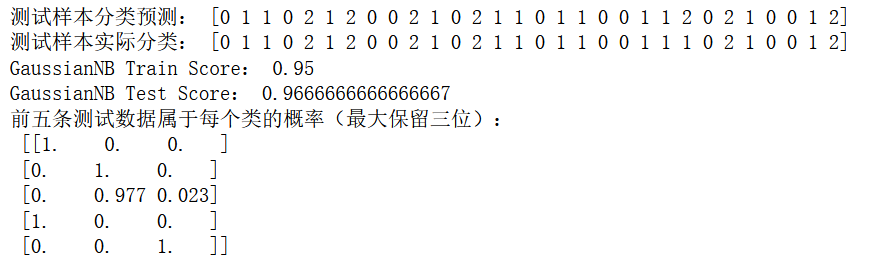


图1 高斯分布的朴素贝叶斯算法模型测试结果

（2）查看训练出模型的各个参数。

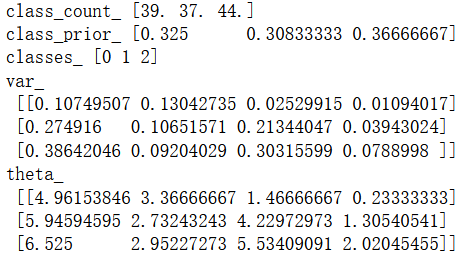


图2 查看模型的参数

## 3.3 伯努利分布的朴素贝叶斯算法

### 3.3.1 原理

由于伯努利分布的朴素贝叶斯算法（BernoulliNB）假设特征的先验概率为二元伯努利分布，所以BernoulliNB一般用于样本特征为二元离散值或很稀疏的多元离散值的数据的分类问题。比如，以特征是否出现（0代表没出现，1代表出现）进行统计计算。

### 3.3.2 实验

（1）数据预处理

还是继续使用鸢尾花数据集，但是需要通过一个阈值（threshold）将数据集中样本特征转化为二元离散值，具体步骤为：先对特征数据进行归一化，之后调用sklearn.preprocessing中的binarize(X, \*, threshold)方法进行二元离散化，即大于threshold（0,1之间）的特征为1，小于threshold的特征为0。

也可以在创建伯努利分布的朴素贝叶斯算法实例时，通过binarize参数，自动地在训练模型时将数据二元离散化（见后文）。

本实验将使用binarize参数，所以不需要进行额外的预处理。代码如下：

from sklearn.datasets import load\_iris

from sklearn.naive\_bayes import BernoulliNB

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import binarize

from sklearn.preprocessing import MinMaxScaler

# 导入鸢尾花数据

iris\_data = load\_iris()

data = iris\_data.data

target = iris\_data.target

# 划分数据集

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(

data,

target,

test\_size=0.2,

random\_state=1

)

# 对数据归一化

threshold = 0.5 # 二元离散化阈值

MMS = MinMaxScaler()

MMS.fit(X\_train) # 只在训练集上训练归一化处理器

X\_train = MMS.transform(X\_train)

X\_test = MMS.transform(X\_test)

# 本实验在之后用算法构造函数的binarize参数进行二元离散化

# 故，以下步骤仅供参考，不需要加上。

# X\_train = binarize(X\_train, threshold=threshold)

# X\_test = binarize(X\_test, threshold=threshold)

（2）创建算法模型实例

用sklearn.naive\_bayes模块下的BernoulliNB类可以创建伯努利分布的朴素贝叶斯算法模型实例，其构造器重要参数及含义如下：

**binarize：用于设置将特征二元离散化的阈值，如果不设定，则认为传入的样本的特征数据已是二元离散的。**

**fit\_prior：设置是否根据训练数据计算类别的先验概率，默认为True。如果设置为False，则认为每个类别的先验概率相同，即，用类别总数的倒数作为每个类别的先验概率。**

**class\_prior：可人为指定类别的先验概率，如果不指定，则会自动根据训练数据进行计算。**

本实验创建两个伯努利分布的贝叶斯算法模型，一个接收的是连续的特征数据；另一个通过**binarize指定二元离散化的阈值，接收二元离散的特征数据。其他参数使用默认值，代码如下：**

model\_c = BernoulliNB()

model\_bi = BernoulliNB(binarize=threshold)

（3）训练并测试

BernoulliNB算法实例提供的方法与GaussianNB中类似，不再赘述。对两种模型进行训练和测试，观察是否是二元离散特征数据对伯努利分布的朴素贝叶斯算法的影响。

# 训练

model\_c.fit(X\_train, Y\_train)

model\_bi.fit(X\_train, Y\_train)

# 评分

print("BernoulliNB Train Score(Continuous)\n", model\_c.score(X\_train, Y\_train))

print("BernoulliNB Test Score(Continuous)\n", model\_c.score(X\_test, Y\_test))

print("BernoulliNB Train Score(Binary Discrete)\n", model\_bi.score(X\_train, Y\_train))

print("BernoulliNB Test Score(Binary Discrete)\n", model\_bi.score(X\_test, Y\_test))

（4）查看训练出的参数

可以用parameter\_（参数名加下划线）的形式直接访问训练得到的以下参数：

**class\_count\_：每一个类别中的训练样本数量。**

**class\_log\_prior\_：每个类别先验概率的自然对数（由于概率值小于1，所以这个值小于0）。**

**classes\_：分类器已知的类别标签。**

**feature\_count\_：每个类别中每个特征包含的训练样本数量矩阵f\_c，例如f\_c[1][2]表示属于第1类带有第2个特征的样本数量。**

**feature\_log\_prob\_：在某个类下，各个特征的经验概率组成的矩阵。**

查看训练出的各个参数，代码如下：

# 查看参数

print("class\_count\_", model\_bi.class\_count\_)

print("class\_log\_prior\_", model\_bi.class\_log\_prior\_)

print("classes\_", model\_bi.classes\_)

print("feature\_count\_\n", model\_bi.feature\_count\_)

print("feature\_log\_prob\_\n", model\_bi.feature\_log\_prob\_)

### 3.3.3 实验结果

（1）伯努利分布的朴素贝叶斯算法评价结果

由实验结果可看出，二元离散化的特征数据训练出的模型分类效果更佳。

在对原本连续的鸢尾花特征数据二元离散化过程中，损失了大量信息，导致最终分类效果不如高斯分布的朴素贝叶斯算法。但是如果原本数据就是二元离散的，伯努利朴素贝叶斯算法可能会有更好的效果。

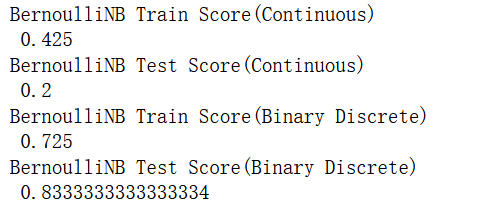


图3 伯努利分布的朴素贝叶斯算法评价结果

（2）查看参数

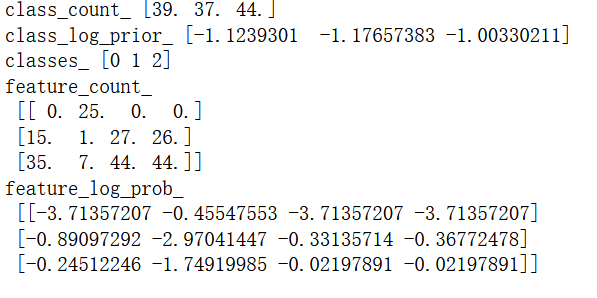


图4 查看模型的参数

## 3.4 多项式分布的朴素贝叶斯算法

### 3.4.1 原理

多项式分布分朴素贝叶斯算法（MultinomialNB）假设特征的先验概率为多项式分布，所以一般用于样本特征为多元离散值的数据。另外，由于多项式实验中的实验结果都很具体，它所涉及的特征往往是次数、频次、计数，出现与否这样的概念，这些概念都是离散的正整数，因此，sklearn中的多项式朴素贝叶斯不接受负值的输入。如，文本分类算法中，会用单次出现次数进行统计计算。

### 3.4.2 实验

（1）数据预处理

继续使用鸢尾花数据集。需要先对数据集进行预处理，将数据通过归一化变换到[0,1]上，接着利用sklearn.preprocessing库中的KbinsDiscretizer，将样本特征转化为k元离散的数据。

用sklearn.preprocessing模块的KbinsDiscretizer类可以创建多值离散化转换器实例，KbinsDiscretizer可以将连续数据分成多个间隔，其重要参数如下：

**n\_bins：指定要将连续变量离散化为多少元。n\_bins需要大于等于2.**

**encode：指定编码转换的方式，默认为“onehot”表示用热编码对转换后的结果进行编码，然后返回一个稀疏矩阵；可选“onehot-dense”，表示**用热编码对转换后的结果进行编码，并返回一个密集数组；或选“**ordinal**”返回编码为整数值的bin标识符。

**strategy：用于定义离散化间隔大小宽度的策略。“uniform”表示保证段间隔都有一样的大小；“quantile”表示保证每段间隔中有一样的样本数。**

**数据预处理的代码如下：**

from sklearn.naive\_bayes import MultinomialNB

from sklearn.datasets import load\_iris

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import KBinsDiscretizer

from sklearn.preprocessing import MinMaxScaler

# 导入鸢尾花数据

iris\_data = load\_iris()

data = iris\_data.data

target = iris\_data.target

# 划分数据集

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(

data,

target,

test\_size=0.2,

random\_state=1

)

# 先对数据归一化，然后把样本特征改成k元离散的数据

MMS = MinMaxScaler()

MMS.fit(X\_train)

X\_train = MMS.transform(X\_train)

X\_test = MMS.transform(X\_test)

kbs = KBinsDiscretizer(n\_bins=6)

kbs.fit(X\_train)

X\_train = kbs.transform(X\_train)

X\_test = kbs.transform(X\_test)

（2）创建训练算法模型并评估

用sklearn.naive\_bayes模块下的MultinomialNB类可以创建多项式分布的朴素贝叶斯算法模型实例，与伯努利分布的朴素贝叶斯算法模型类似，这里不再赘述。代码如下：

# 创建实例

model = MultinomialNB()

# 训练

model.fit(X\_train, Y\_train)

# 评分

print("MultinomialNB Train Score", model.score(X\_train, Y\_train))

print("MultinomialNB Test Score", model.score(X\_test, Y\_test))

### 3.4.3 实验结果

训练出的多项式分布的朴素贝叶斯算法模型评分如下：



图5 多项式分布的朴素贝叶斯算法模型评价结果

# 4 总结

在使用朴素贝叶斯分布进行分类问题时，应该根据样本特征的数据形式来选择合适的朴素贝叶斯算法模型。

一般来说：GaussianNB适用于样本特征为连续值的数据；MultinomialNB适用于样本特征为多元离散值的数据；BernoulliNB适用于样本特征为二元离散值或很稀疏的多元离散值的数据。