HW1-朴素贝叶斯分类法

崔孝荣　何嘉欣

March 2019

1　引言

分类是机器学习中一个重要的模块，应用于现实生活中很多场景，如垃圾邮件的删选、降雨预测。贝叶斯分类是一类分类算法的总称，这类算法均以贝叶斯定理为基础，故统称为贝叶斯分类。其中最为简单的一种算法是朴素贝叶斯分类，本文首先将介绍贝叶斯分类的基础贝叶斯定理，以及贝叶斯决策论，然后将介绍朴素贝叶斯分类原理，最后将以一个实际例子解释朴素贝叶斯分类的具体操作。

2　贝叶斯定理

设样本空间为 S,A 为 E 的事件，$\{B_1, B_2, B_3, ..., B_n\}$ 为 S 的一个划分，且

$$P(B_i) > 0, i = \{1, 2, 3, ..., n\}$$

那么：

$$P(B_i|A) = \frac{P(A|B_i)P(B_i)}{\sum_{j=1}^{n} P(A|B_j)P(B_j)}, i = \{1, 2, 3, ..., n\}.$$

3　贝叶斯决策论

贝叶斯决策是在概率论的框架下对样本进行分类的方法，在所有相关概率都已知的理想情形下，贝叶斯决策考虑基于这个概率和误判损失来对样本进行分类。设有 N 种可能的类别，即 $Y = \{c_1, c_2, c_3, ..., c_n\}$,$\lambda_{ij}$ 是将 c_j 误分类为 c_i 的误差损失。基于后验概率 $P(c_i|\mathbf{x})$ 可获得将样本 $\mathbf{x}$ 误分类为 ci 所产生的期望损失（条件风险）：

$$R(c_i|\mathrm{x}) = \sum_{j=1}^{n} \lambda_{ij} P(c_j|\mathbf{x})$$

我们需要寻找一个判定准则 h, 使得总体风险最小,

$$R(h) = E_{\mathrm{x}}[R(h(\mathrm{x})|\mathrm{x})]$$

也就是说我们只需要找到 h，使得样本 $\mathbf{x}$ 的期望损失最小，那么总体风险也会最小，这就是贝叶斯决策论，此时

$$h^* = argmin_{c\in\ y} R(c|x)$$

被称为贝叶斯最优分类器。一般来说，误判损失 λ_{ij} 可以写作

$$\lambda_{ij} = \begin{cases} 1 & i \neq j \\ 0 & i = j \end{cases}$$

也就是说判断正确的损失为 0，判断错误的损失为 1，这个时候条件风险

$$R(c|\mathrm{x}) = 1 - P(c|\mathbf{x})$$

此时的贝叶斯最优分类器为

$$h^* = argmin_{c\in\ y} R(c|x)$$

也就是对每个样本选择后验概率 P(c|x) 最大的类别标记，也即选择最有可能的类别。

根据贝叶斯定理有：

$$P(c|x) = \frac{P(x|c)P(c)}{P(x)}$$

其中 P(c) 是先验概率，P(x|c) 是样本 x 相对与类别 c 的类条件概率，这个时候，最有问题就转化为如何从训练样本中获取 P(c) 和 P(x|c)。

4 朴素贝叶斯

朴素贝叶斯分类是一种十分简单的分类算法，采用了“属性条件独立性假设”，也即是假设每个属性独立的对分类结果发生影响。朴素贝叶斯的思想基础是这样的：对于给出的待分类项，求解在此项出现的条件下各个类别出现的概率，哪个最大，就认为此待分类项属于哪个类别。朴素贝叶斯分类的正式定义如下：

1、设 $x = \{a_1, a_2, ..., a_m\}$ 为一个待分类项，而每个 a 为 x 的一个特征属性。

2、有类别集合 $C = \{y_1, y_2, ..., y_n\}$。

3、计算 $P(y_1|x), P(y_2|x), ..., P(y_n|x)$。

4、如果 $P(y_k|x) = max\{P(y_1|x), P(y_2|x), ..., P(y_n|x)\}$，则 $x \in y_k$。

那么现在的关键就是如何计算第 3 步中的各个条件概率。我们可以这么做：

1、找到一个已知分类的待分类项集合，这个集合叫做训练样本集。

2、统计得到在各类别下各个特征属性的条件概率估计。即

$$P(a_1|y_1), P(a_2|y_1), ..., P(a_m|y_1); P(a_1|y_2), P(a_2|y_2), ..., P(a_m|y_2); ...P(a_1|y_n), P(a_2|y_n), ..., P(a_m|y_n).$$

3、如果各个特征属性是条件独立的，则根据贝叶斯定理有如下推导：

$$P(y_i|x) = \frac{P(x|y_i)P(y_i)}{P(x)}$$

因为分母对于所有类别为常数，因为我们只要将分子最大化皆可。又因为各特征属性是条件独立的，所以有：

$$P(x|y_i)P(y_i) = P(a_1|y_i)P(a_2|y_i)...P(a_m|y_i)P(y_i) = P(y_i)\prod_{j=1}^{m} P(a_j|y_i)$$

变量名	解释
$gender$	$object$
$race/ethnicity$	$object$
$parentallevelofeducation$	$object$
$lunch$	$object$
$testpreparationcourse$	$object$
$mathscore$	$int64$
$readingscore$	$int64$
$writingscore$	$int64$

表 1: 学生成绩数据变量解释

5 examples

这个例子的数据集为 Students Performance in Exams[1]，变量解释如表 1

在这个例子里，1000 个学生为一个待分类项，而学生性别、民族、家长的受教育程度、是否吃午饭等都是他的一个特征属性，我们选取 parental level of education 及 lunch 作为待分类项，类别集合为 math score。运用 python 里面 sklearn 的安装包就能够完成一个朴素的贝叶斯回归。1. 找到训练样本集 StudentsPerformance.csv 2. 运用 Python 统计得到各个特征属性的条件概率估计 3. 求出分子最大的特征。Python 输出结果如图 1，图 2。[2]

参考文献

[1] 周志华. 机器学习 [M]. 清华大学出版社, 2016.

[2] 盛骤等编. 概率论与数理统计 [M]. 高等教育出版社, 1989.

[3] Andreas C. Müller and Sarah Guido. Introduction to Machine Learning with Python: A Guide for Data Scientists[M].O'Reilly Media, 2016.

[1]数据集的下载地址及详细介绍为：https://www.kaggle.com/spscientist/students-performance-in-examsStudentsPerformance.csv

[2]python 操作参考：Scikit-learn Tutorial: Machine Learning in Python，网址为：https://www.dataquest.io/blog/sci-kit-learn-tutorial；及知乎专栏：Sklearn 参数详解—贝叶斯，网址为：https://zhuanlan.zhihu.com/p/39780650

```python
# import necessary modules
import pandas as pd
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
# store the data in a variable
data190315 = open("C:\\Users\\Xin\\PycharmProjects\\untitled\\StudentsPerformance.csv")
# Read in the data with `read_csv()`
S_data = pd.read_csv(data190315, encoding='utf-8')
# Using .head() method to view the first few records of the data set
print(S_data.head())
# using the dtypes() method to display the different datatypes available
print(S_data.dtypes)
# import the necessary module
from sklearn import preprocessing

# create the Labelencoder object
le = preprocessing.LabelEncoder()

print("gender : ", S_data['gender'].unique())
print("race/ethnicity : ", S_data['race/ethnicity'].unique())
print("parental level of education : ", S_data['parental level of education'].unique())
print("lunch : ", S_data['lunch'].unique())
print("test preparation course : ", S_data['test preparation course'].unique())
```

```python
# convert the categorical columns into numeric
S_data['gender'] = le.fit_transform(S_data['gender'])
S_data['race/ethnicity'] = le.fit_transform(S_data['race/ethnicity'])
S_data['parental level of education'] = le.fit_transform(S_data['parental level of education'])
S_data['lunch'] = le.fit_transform(S_data['lunch'])
S_data['test preparation course'] = le.fit_transform(S_data['test preparation course'])

# select columns other than 'race/ethnicity','math score', 'test preparation course', 'reading score' and 'writing score'
cols = [col for col in S_data.columns if
        col not in ['race/ethnicity', 'math score', 'test preparation course' 'reading score', 'writing score', 'gender']]
# dropping the 'race/ethnicity', 'reading score', 'writing score', 'gender' and 'math score' columns
data = S_data[cols]
ms = S_data['math score']
print(ms)
for index in ms.index:
    if ms[index] < 60:
        ms[index] = 0
    elif 60 <= ms[index] < 70:
        ms[index] = 1
    elif 70 <= ms[index] < 80:
        ms[index] = 2
    elif 80 <= ms[index] < 90:
        ms[index] = 3
    else:
        ms[index] = 4
```

图 1: 朴素贝叶斯 Python 代码

```
# assigning the 'ms' column as target
target = ms

print(data.head(n=2))
print(ms.head(n=5))

# split data set into train and test sets
data_train, data_test, target_train, target_test = train_test_split(data, target, test_size=0.20, random_state=10)
print(data_train, data_test, target_train, target_test)


# create an object of the type MultinomialNB
mtn = MultinomialNB(alpha=1.0, fit_prior=True, class_prior=None)



# train the algorithm on training data and predict using the testing data
pred = mtn.fit(data_train, target_train).predict(data_test)
pred1 = mtn.fit(data_train, target_train).predict_proba(data_test)

print("\n==Predict result by multinomial predict==")
print(pred.tolist())
print("\n==Predict result by multinomial predict_proba==")
print(pred1.tolist())

# print the accuracy score of the model
print("Multinomial Naive-Bayes accuracy : ", accuracy_score(target_test, pred, normalize=True))

# create an object of the type GaussianNB
gnb = GaussianNB()

#train the algorithm on training data and predict using the testing data
Gau = gnb.fit(data_train, target_train).predict(data_test)
Gau1 = gnb.fit(data_train, target_train).predict_proba(data_test)
print("\n==Predict result by multinomial predict==")
print(Gau.tolist())
print("\n==Predict result by multinomial predict_proba==")
print(Gau1.tolist())

#print the accuracy score of the model
print("Gaussian Naive-Bayes accuracy : ",accuracy_score(target_test, pred, normalize = True))
```

图 2: 朴素贝叶斯 Python 代码

```
==Predict result by multinomial predict==
[0, 0, 0, 1, 0, 1, 3, 0, 1, 3, 3, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0,

==Predict result by multinomial predict_proba==
[[0.5360370689873948, 0.2562817374385647, 0.15487890729675105, 0.04864137283388286,
Multinomial Naive-Bayes accuracy :  0.325

==Predict result by multinomial predict==
[0, 3, 1, 1, 2, 0, 3, 1, 0, 4, 3, 0, 3, 1, 1, 1, 3, 3, 1, 1, 3, 1, 1, 2, 1, 3, 2, 1,

==Predict result by multinomial predict_proba==
[[0.5760299203655193, 0.35112861591881084, 0.06416652554600699, 0.008612465485118416
Gaussian Naive-Bayes accuracy :  0.325
```

图 3: 朴素贝叶斯分类结果（节选）