利用朴素贝叶斯对名字进行性别预测

3个小节，预计用时**30分钟**。

请打开您的电脑，按照步骤一步步完成哦！

本教程基于**Python 3.5**。

原创者：**s3040608090 (http://sofasofa.io/user_competition.php?id=1001216)** | 修改校对：SofaSofa TeamC |

---

1. 条件概率与贝叶斯定理

对于事件A和B，当B发生的情况下，A发生的条件概率为

$$P(A|B) = \frac{P(AB)}{P(B)}.$$

如果把$P(AB)$表示为$P(B|A)P(A)$，那么

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}.$$

2. 朴素贝叶斯

朴素贝叶斯是一个基于贝叶斯定理的分类算法，其基本假设是所有特征是相互独立的。 举个例子来说，有一个二元分类问题，每个样本只有两个二元特征X_1和X_2。若已知一个样本$(X_1 = 1, X_2 = 0)$，我们要预测它的标签为1的概率，就是等价于去计算

$$P(Y = 1|X_1 = 1, X_2 = 0)$$

根据贝叶斯定理，我们可得

$$P(Y = 1|X_1 = 1, X_2 = 0) = \frac{P(Y = 1)P(X_1 = 1, X_2 = 0|Y = 1)}{P(X_1 = 1, X_2 = 0)}$$

其中$P(Y = 1)$被称为先验（prior），$P(X_1 = 1, X_2 = 0|Y = 1)$被称为似然（likelyhood），$P(X_1 = 1, X_2 = 0)$被成为证据（evidence）。

因为我们假设所有特征独立，所以我们可以把$P(Y = 1|X_1 = 1, X_2 = 0)$写成

$$P(Y = 1|X_1 = 1, X_2 = 0) = \frac{P(Y = 1)P(X_1 = 1|Y = 1)P(X_2 = 0|Y = 1)}{P(X_1 = 1)P(X_2 = 0)}$$

推广到更普遍的情况下，假设数据有k个特征，

$$P(Y|X_1, X_2, \cdots, X_n) = \frac{1}{Z}P(Y)\prod_{i=1}^{n}P(X_i|Y)$$

其中Z是缩放因子，使得概率和为1。

对于一个分类问题，如果我们只需要得到其标签，我们只需要求解

$$y_{pred} = \arg\max_y P(Y=y)\prod_{i=1}^{n} P(X_i|Y=y)$$

3. 实战练习

下面我们利用朴素贝叶斯对“**机器读中文：根据名字判断性别 (http://sofasofa.io/competition.php?id=3)**”中的数据进行预测。首先下载，并读取数据。

```
# -*- coding: utf-8 -*-

import pandas as pd
from collections import defaultdict
import math

# 读取train.txt
train = pd.read_csv('train.txt')
test = pd.read_csv('test.txt')
submit = pd.read_csv('sample_submit.csv')
```

看看训练集中的数据长什么样

```
train.head(10)
```

	id	name	gender
0	1	闵家	1
1	2	玉瓔	0
2	3	于邺	1
3	4	越英	0
4	5	蕴萱	0
5	6	子颀	0
6	7	靖曦	0
7	8	鲁莱	1
8	9	永远	1
9	10	红孙	1

```
# 把数据分为男女两部分
names_female = train[train['gender'] == 0]
names_male = train[train['gender'] == 1]

# totals用来存放训练集中女生、男生的总数
totals = {'f': len(names_female),
          'm': len(names_male)}
```

分别计算在所有女生（男生）的名字当中，某个字出现的频率。这一步相当于是计算 $P(X_i|\text{女生})$和 $P(X_i|\text{男生})$

```
frequency_list_f = defaultdict(int)
for name in names_female['name']:
    for char in name:
        frequency_list_f[char] += 1. / totals['f']

frequency_list_m = defaultdict(int)
for name in names_male['name']:
    for char in name:
        frequency_list_m[char] += 1. / totals['m']
```

```
print(frequency_list_f['娟'])
```

```
0.004144009000562539
```

```
print(frequency_list_m['钢'])
```

```
0.0006299685015749209
```

上面两个例子说明$P(\text{名字中含有娟}|\text{女生}) = 0.004144$，$P(\text{名字中含有钢}|\text{男生}) = 0.0006299$

考虑到预测集中可能会有汉字并没有出现在训练集中，所以我们需要对频率进行Laplace平滑（什么是Laplace平滑 (http://sofasofa.io/forum_main_post.php?postid=1001239)）。

```
def LaplaceSmooth(char, frequency_list, total, alpha=1.0):
    count = frequency_list[char] * total
    distinct_chars = len(frequency_list)
    freq_smooth = (count + alpha ) / (total + distinct_chars * alpha)
    return freq_smooth
```

回顾第2节中的式子

$$P(Y)\prod_{i=1}^{n}P(X_i|Y),$$

在性别预测中，每个样本中大量的特征都是0。比如说只有$X_2 = 1$，其他都为0，那么

$$y_{pred} = \arg\max_y P(Y = y)P(X_2 = 1|Y = y)\frac{\prod_{i=1}^{n} P(X_i = 0|Y = y)}{P(X_2 = 0|Y = y)}$$

由于$P(X_i)$的数值通常较小，我们对整体取对数（防止浮点误差），可得

$$\log P(Y = y) + \sum_{i=1}^{n} \log P(X_i = 0|Y = y) + \log P(X_2 = 1|Y = y) - \log P(X_2 = 0|Y = y)$$

如果一个人的名字中有两个字，假设$X_5 = 1$，$X_{10} = 1$，其余为0，那么该名字的对数概率表达式为

$$\log P(Y = y) + \sum_{i=1}^{n} \log P(X_i = 0|Y = y)$$

$$+ \log P(X_5 = 1|Y = y) - \log P(X_5 = 0|Y = y) + \log P(X_{10} = 1|Y = y) - \log P(X_{10} = 0|Y = y)$$

对于一种性别，$\log P(Y = y) + \sum_{i=1}^{n} \log P(X_i = 0|Y = y)$只需要计算一次。为了方面，我们将其数值存放在bases当中

```
base_f = math.log(1 - train['gender'].mean())
base_f += sum([math.log(1 - frequency_list_f[char]) for char in frequency_l
ist_f])

base_m = math.log(train['gender'].mean())
base_m += sum([math.log(1 - frequency_list_m[char]) for char in frequency_l
ist_m])

bases = {'f': base_f, 'm': base_m}
```

对于$\log P(X_i = 1|Y) - \log P(X_i = 0|Y)$部分，我们利用如下函数计算

```
def GetLogProb(char, frequency_list, total):
    freq_smooth = LaplaceSmooth(char, frequency_list, total)
    return math.log(freq_smooth) - math.log(1 - freq_smooth)
```

最后我们只需要组合以上函数，实现

$$y_{pred} = \arg\max_y P(Y = y)P(X_2 = 1|Y = y)\frac{\prod_{i=1}^{n} P(X_i = 0|Y = y)}{P(X_2 = 0|Y = y)}$$

```
def ComputeLogProb(name, bases, totals, frequency_list_m, frequency_list_f
):
    logprob_m = bases['m']
    logprob_f = bases['f']
    for char in name:
        logprob_m += GetLogProb(char, frequency_list_m, totals['m'])
        logprob_f += GetLogProb(char, frequency_list_f, totals['f'])
    return {'male': logprob_m, 'female': logprob_f}

def GetGender(LogProbs):
    return LogProbs['male'] > LogProbs['female']

result = []
for name in test['name']:
    LogProbs = ComputeLogProb(name, bases, totals, frequency_list_m, freque
ncy_list_f)
    gender = GetGender(LogProbs)
    result.append(int(gender))

submit['gender'] = result

submit.to_csv('my_NB_prediction.csv', index=False)
```

最后结果输出在 'my_NB_prediction.csv' 中。不如上传到比赛页面 (http://sofasofa.io/competition.php?id=3)看看结果哦。

我们可以看看预测结果如何。

```
test['pred'] = result
test.head(20)
```

	id	name	pred
0	0	辰君	0
1	1	佳遥	0
2	2	淼剑	1
3	3	浩苳	1
4	4	俪妍	0
5	5	秉毅	1
6	6	妍艺	0
7	7	海防	1
8	8	壬尧	1

	id	name	pred
9	9	珞千	0
10	10	义元	1
11	11	才君	1
12	12	吉喆	1
13	13	少竣	1
14	14	创海	1
15	15	熙兰	0
16	16	家冬	1
17	17	方荧	1
18	18	介二	1
19	19	钰泷	1

完整代码如下：

```
# -*- coding: utf-8 -*-

import pandas as pd
from collections import defaultdict
import math

# 读取train.txt
train = pd.read_csv('train.txt')
test = pd.read_csv('test.txt')
submit = pd.read_csv('sample_submit.csv')

#把数据分为男女两部分
names_female = train[train['gender'] == 0]
names_male = train[train['gender'] == 1]

totals = {'f': len(names_female),
          'm': len(names_male)}

frequency_list_f = defaultdict(int)
for name in names_female['name']:
    for char in name:
        frequency_list_f[char] += 1. / totals['f']

frequency_list_m = defaultdict(int)
for name in names_male['name']:
    for char in name:
        frequency_list_m[char] += 1. / totals['m']

def LaplaceSmooth(char, frequency_list, total, alpha=1.0):
    count = frequency_list[char] * total
    distinct_chars = len(frequency_list)
    freq_smooth = (count + alpha ) / (total + distinct_chars * alpha)
    return freq_smooth

def GetLogProb(char, frequency_list, total):
    freq_smooth = LaplaceSmooth(char, frequency_list, total)
    return math.log(freq_smooth) - math.log(1 - freq_smooth)

def ComputeLogProb(name, bases, totals, frequency_list_m, frequency_list_f
):
    logprob_m = bases['m']
    logprob_f = bases['f']
    for char in name:
        logprob_m += GetLogProb(char, frequency_list_m, totals['m'])
        logprob_f += GetLogProb(char, frequency_list_f, totals['f'])
    return {'male': logprob_m, 'female': logprob_f}

def GetGender(LogProbs):
    return LogProbs['male'] > LogProbs['female']


base_f = math.log(1 - train['gender'].mean())
```

```
base_f += sum([math.log(1 - frequency_list_f[char]) for char in frequency_l
ist_f])

base_m = math.log(train['gender'].mean())
base_m += sum([math.log(1 - frequency_list_m[char]) for char in frequency_l
ist_m])

bases = {'f': base_f, 'm': base_m}


result = []
for name in test['name']:
    LogProbs = ComputeLogProb(name, bases, totals, frequency_list_m, freque
ncy_list_f)
    gender = GetGender(LogProbs)
    result.append(int(gender))

submit['gender'] = result

submit.to_csv('my_NB_prediction12.csv', index=False)
```