本节课讲解第三个分类算法——朴素贝叶斯，我依然以一个例子开头，带领你进入朴素贝叶斯算法的世界，通过算法原理、算法优缺点的讲解，带你算一算是否要买航空延误险。最后我们再动手来写一下代码，看看如何使用朴素贝叶斯来进行分类。

一个例子

最近看到一则新闻，王女士从 2015 年开始，凭借自己对航班和天气的分析，成功地购买了大约 900 次飞机延误险并获得延误赔偿，累计获得保险理赔金高达 300 多万元。那么她是怎么决定要买延误险的呢？

其实，航班延误最主要的原因就是天气变化，包括起飞地及降落地的天气；除此之外，也有机场和航空公司的原因。假设这些原因之间并没有互相影响，每一项对于飞机最终是否延误的影响都是独立的，王女士集齐过去的数据，就可以计算出每一个条件与飞机延误的概率。比如，在总体上延误的概率为 20%，不延误的概率为 80%。在飞机延误的情况下，“起飞地天气 = 晴天” 的概率为 20%，“降落地天气 = 雨天” 的概率为 40%，“机场 = 首都机场” 的概率为 35%，“航空公司 = 南方航空” 的概率为 5%；在不延误的情况下，这些属性的概率分别为 60%、55%、45%、55%。

那么这个时候，有一架南方航空公司的航班，从北京飞往上海，北京天气是晴天，上海天气是雨天，那么，我们就可以根据上面的概率算出来不延误的综合概率 = 80% × 60% × 55% × 45% × 55% = 0.0065412，延误的综合概率 = 20% × 20% × 40% × 35% × 5% = 0.00028，从这个结果来看，不延误的可能性要高于延误的可能性，所以这次不需要买延误险。

算法原理

上面的这个例子就是我今天要介绍的算法——朴素贝叶斯分类器的思路，贝叶斯这个名字你应该很熟悉，这简直就是概率论的鼻祖，所以我们这个算法的原理也跟概率论脱不开干系。考虑我们分类所用到的特征和分类结果，朴素贝叶斯有一个假设前提，那就是所有的条件对结果都是独立发生作用的。就像我们上面预测是否要买延误险一样，起飞地的天气不会对降落地的天气有影响，同时起飞地和降落地的天气以及所造成的延误问题不会比他们单独发生时有任何区别。所以根据这个思想，出现了朴素贝叶斯概率公式：

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

这个公式看不懂没关系，我把它转换成我们的属性和分类，再来看一下：

$$\mathbf{P}(\text{分类}|\text{属性}) = \frac{P(\text{属性}|\text{分类})P(\text{分类})}{P(\text{属性})}$$

在某些属性已知的情况下，获得某个分类的概率就等于在已知分类的情况下，某个属性的概率乘以分类的概率再除以属性的概率。具体到我们的例子上，如下所示：

$$P(\text{延误}|\text{起飞地=晴天，将落地=雨天，机场=首都机场，航空公司=南方航空}) =$$

$$\frac{P(\text{起飞地=晴天，将落地=雨天，机场=首都机场，航空公司=南方航空}|\text{延误})P(\text{延误})}{P(\text{起飞地=晴天，将落地=雨天，机场=首都机场，航空公司=南方}$$

我们需要获得的是左侧的结果，而右侧就是基于已有的样本数据可以计算出来。而根据上面提到的朴素贝叶斯的假设，最重要的就是下面这个转换：

P(起飞地=晴天，将落地=雨天，机场=首都机场，航空公司=南方航空|延误)=

P(起飞地=晴天|延误)*P(将落地=雨天|延误)*P(机场=首都机场|延误)*P(航空公司=南

于是，根据已经有的数据，我们可以计算出每一个特征对最终结果的概率情况，比如，有 100 条数据，结果延误的有 20 条，结果不延误的有 80 条。在延误的 20 条中，起飞地 = 晴天的有 4 条，那么在已知延误的情况下起飞地是晴天的概率为 20%；在不延误的 80 条数据中，起飞地 = 晴天的有 48 条。那么在已知不延误的情况下，起飞地是晴天的概率为 60%。依据这个方法，我们就可以计算出所有我们需要的概率值，像上面的例子中那样。

如何处理连续值

预测购买延误险的例子中使用的都是离散值，那么对于连续值该怎么处理呢？假设我们新增一个特征——机票的价格。这时，机票价格是一个连续值，我们可以假设机票这个特征服从正态分布，通过样本集计算出机票价格对应每一个分类的均值和方差，再根据比如密度函数，计算出新数据与均值的距离，从而获得一个概率值。关于这一块的处理细节，如果你有兴趣可以再查找一些详细的讲解。

关于平滑

对于离散值，有一个需要注意的地方：如果某一个属性值比如 “航空公司 = 西藏航空”，由于数据比较少，在“分类 = 延误” 的类别下没有出现过，那么按照上面的方法，不管其他的特征如何，P （“航空公司 = 西藏航空”| 延误）= 0，所有用到这个的结果都会是 0。那么，这里就有一个数据准备环节的方法，称为“平滑”（Smoothing）。这时我们可以想到一个简单的平滑方案，那就是在分子分母中同时加上一个较小的值，来防止出现分子或分母为 0 的情况。除了我说的这种简便方法外，还有常用的拉普拉斯修正、古德 - 图灵估计法、Katz 等平滑的方法，这里就不一一介绍了。

已经清楚了算法的基本原理，接下来我们再看一下朴素贝叶斯算法有什么优缺点。

算法的优缺点

优点

- **逻辑清晰简单、易于实现，适合大规模数据。** 根据算法的原理，只要我们把样本中所有属性相关的概率值都计算出来，然后分门别类地存储好，就获得了我们的朴素贝叶斯模型。
- **运算开销小。** 根据上一条我们可以得知，所有预测需要用到的概率都已经准备好，当新数据来了之后，只需要获取对应的概率值，并进行简单的运算就能获得结果。
- **对于噪声点和无关属性比较健壮。** 噪声点和无关属性对算法影响较小，在很多邮件服务中仍然一直沿用这个方法进行垃圾邮件过滤。
- **预测过程快。** 由于所有需要用到的属性相关概率都已经计算出来了，在新数据到来需要预测的时候，只需要把对应的一些概率值取出来进行计算就可以获得结果，所用到的时间和空间都很小。

缺点

由于做了 “各个属性之间是独立的” 这个假设，同样该算法也暴露了缺点。因为在实际应用中，属性之间完全独立的情况是很少出现的，如果属性相关度较大，那么分类的效果就会变差。所以在具体应用的时候要好好考虑特征之间的相互独立性，再决定是否要使用该算法，比如，维度太多的数据可能就不太适合，因为在维度很多的情况下，不同的维度之间越有可能存在联合的情况，而不是相互独立的，那么模型的效果就会变差。

尝试动手

通过案例我们了解了算法的原理及其优缺点，下面动动手指尝试使用贝叶斯算法来解决问题吧。

本节的代码主要调整的部分就是引入的算法模块，其他部分在前面的实践中都已经写过，你实现起来应该已经轻车熟路了。

```
from sklearn import datasets

from sklearn.naive_bayes import GaussianNB

import numpy as np

np.random.seed(0)

iris=datasets.load_iris()

iris_x=iris.data

iris_y=iris.target

indices = np.random.permutation(len(iris_x))

iris_x_train = iris_x[indices[:-10]]

iris_y_train = iris_y[indices[:-10]]

iris_x_test  = iris_x[indices[-10:]]

iris_y_test  = iris_y[indices[-10:]]

clf = GaussianNB()

clf.fit(iris_x_train,iris_y_train)

iris_y_predict = clf.predict(iris_x_test)

score=clf.score(iris_x_test,iris_y_test,sample_weight=None)

print('iris_y_predict = ')

print(iris_y_predict)
```

```
print('iris_y_test = ')


print(iris_y_test)


print('Accuracy:',score)
```

我们来看一下代码的输出结果，可以看到，这次仍然是第二个没有预测正确，模型的准确率为 90%。

```
iris_y_predict =


[1 2 1 0 0 0 2 1 2 0]


iris_y_test =


[1 1 1 0 0 0 2 1 2 0]


Accuracy: 0.9
```

这次的代码实践没有增加太多新的内容，你写起来应该非常轻松了。在实际的研究中，也有不少基于朴素贝叶斯算法发展出来的新方法，下面我们再来看一下朴素贝叶斯的扩展内容。

扩展内容

前面我们讲到，朴素贝叶斯是基于属性之间的独立性假设而进行构建的，但是实际上很多时候这种假设存在一定的局限，为了打破这些局限，于是衍生出了一些方法，下面我列举两个：

- 半朴素贝叶斯 ODE（One Dependent Estimator）：为了解决朴素贝叶斯中属性独立性假设在实际生活中不太适用的问题，新的研究方案尝试建立一些属性间的联系，假定属性有一定的相关性，从而产生的算法被称为**半朴素贝叶斯方法。**这里说的 ODE 就是其中的一种，在 ODE 中，假设每个属性最多依赖于一个其他的属性，这样修正了一些有依赖的情况，同时又没有增加太多的计算复杂性。
- AODE（Averaged One Dependent Estimator）：AODE 其实是一种集成学习的方法。在 ODE 的基础上，使用 bagging 集成学习的思路，训练多个模型，其中每个模型都设置一种属性作为其他所有属性的关联属性，最后使用这多个模型的结果平均数值作为最终结果。

总结

讲到这里，我们这一课时的内容又要结束了，不知道你是否意犹未尽？我在这节课里讲解的朴素贝叶斯算法是一个非常简洁的算法，只需要进行比较简单的数学计算就可以获得我们所要的结果。在开头，我列举了一个关于计算是否要买延误险的例子，那么通过这一节的学习，你下次买机票的时候是不是也可以算一算自己是不是要买延误险了？这节课的动手实践部分比较简单，希望能够加深你的动手能力。