# 数学与科学的反映-贝式网络

李御玺([leeys@mail.mcu.edu.tw](mailto:leeys@mail.mcu.edu.tw))

## 简介贝式网络 (BAYES NET)

• 朴素贝叶斯(贝式网络的简化)

朴素贝叶斯可以认为是贝式网络的简化, 简化的地方在于它认为输入的字段彼此之间是独立的, 即它认为输入字段和输入字段之间是没有关联的. 朴素贝叶斯做这个简化的目的就是简化计算, 加速计算.

• 贝式网络

• 贝式定理

贝式网络和朴素贝叶斯都是基于概率论中的贝式定理对概率进行估计的

推测在X已知的情况下Y的概率值, 即条件概率. 等于X和Y同时出现的概率除以X单独出现的概率.

即在X已知的情况下Y的概率等于Y已知的情况下X的概率乘以Y的概率除以X的概率

## 朴素贝叶斯 (NAIVE BAYES)

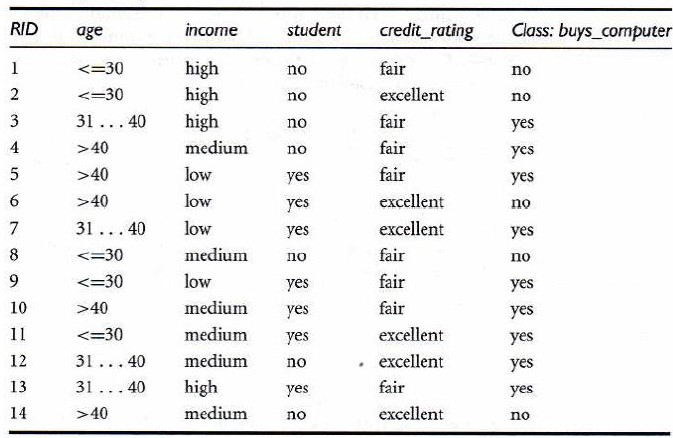
Naive Bayesian Classification

朴素贝叶斯分类推估的过程

**案例: 是否会买电脑**

有一个训练数据集, 一共有14笔数据, 前4个字段为输入字段, age, income, student, credit\_rating. 目标字段为buys\_computer, 即最后会不会购买电脑.

想要使用朴素贝叶斯来推估一个某个人会不会购买电脑.



X为要检测的人的4个输入字段的值, 可以认为这个人是测试数据集中的一个数据.

Classify X=(age=“<=30”, income=“medium”, student=“yes”, credit-rating=“fair”)

要使用朴素贝叶斯来计算这个人会购买电脑的概率和不购买电脑的概率, 就要先计算出 P(buys\_computer=yes | X) 和 P(buys\_computer=no | X). 也就是在X中4个条件的情况下这个人购买电脑的概率和在X中的4个条件下这个人不购买电脑的概率. 如果计算到的会买电脑的概率大于不会买电脑的概率, 就预测这个人会购买电脑, 反之就预测此人不会购买电脑.

**P(Y|X) = P(X|Y) \* P(Y) / P(X) Bayes Theorem**

根据贝氏定理, X已知的情况下Y发生的概率等于Y已知情况下X发生的概率乘以Y发生的概率除以X发生的概率.

因为最终的结果是要比较 P(buys\_computer=yes|X)和P(buys\_computer=no|X)哪个值大, 所以除不除以P(X)并不影响两个值的大小. 可以认为

这两个值的结果是来自于训练数据集, 在对人没有任何限制的条件下, 一个人购买电脑的概率和不买电脑的概率分别为:

P(buys\_computer=yes) = 9/14

P(buys\_computer=no)=5/14

下面要计算在买电脑的条件下出现X中情况的概率和不买电脑的前提条件下出现X中情况的概率.

P(X | buys\_computer=yes)

P(X | buys\_computer=no)

由于情况比较复杂, 可以把问题拆分为4个问题. 所求的P(X|buys\_computer=yes)的概率就等于4个 yes 的概率的乘积, 所求的P(X|buys\_computer=no)等于4个 no 的概率的乘积

1. 在买电脑/不买电脑的情况下年龄小于等于30岁的概率

P(age=<30 | buys\_computer=yes) = 2/9

P(age=<30 | buys\_computer=no) = 3/5

2. 在买电脑/不买电脑的情况下收入为中等水平的概率

P(income=medium | buys\_computer=yes) = 4/9

P(income=medium | buys\_computer=no) = 2/5

3. 在买电脑/不买电脑的情况下是学生的概率

P(student=yes | buys\_computer=yes) = 6/9

P(student=yes | buys\_computer=no) = 1/5

4. 在买电脑/不买电脑的情况下信用评级等于fair的概率

P(credit-rating=fair | buys\_computer=yes) = 6/9

P(credit-rating =fair | buys\_computer=no) = 2/5

P(X | buys\_computer=yes) = (2/9)\*(4/9)\*(6/9)\*(6/9)=0.044

P(X | buys\_computer=no) = 0.019

P(buys\_computer=yes|X) = P(X|buys\_computer=yes) \* P(buys\_computer=yes)=0.028

P(buys\_computer=no|X) = P(X|buys\_computer=no) \* P(buys\_computer=no)=0.007

由于0.028大于0.007, 所以就预测这个顾客会买电脑.

使用朴素贝叶斯进行分类的预测时, 不仅要给出结果为Yes或No, 还要给出对应的概率. 但是由于P(X)未知, 无法直接求出. 但知道一个人要么买电脑, 要么不买电脑, 即买电脑与不买电脑的概率之和应该等于1.

P(buys\_computer=yes | X) \* P(X) + P(buys\_computer=no | X) \* P(X) = 1

P(X) \* (0.028 + 0.007) = 1

P(X) = 1/(0.028 + 0.007)

所以买电脑的概率为

P(buys\_computer=yes | X) \* P(X) = 0.028 \* (1/(0.028+0.007)) =

不买电脑的概率为

P(buys\_computer=no | X) \* P(X) = 0.007 \* (1/(0.028+0.007))

以上的过程也就是对概率进行标准化的过程, 也就是说, 概率一定要进行标准化/正规化, 概率加起来等于100%.

第2点需要注意的, 本来是买电脑的4个联合条件, 我们把它拆分为了4个单独的条件, 并把4个单独的条件概率相乘得到联合条件的概率, 这样做成立的条件就是4个输入字段之间是彼此独立的, 没有关系的.

**P(X, Y) = P(X) \* P(Y | X) = P(Y) \* P(X | Y)= P(X) \* P(Y) if X & Y are independent**

但对4个字段进行考查, 年龄和收入是否是不相关的, 不是的, 一般情况下年龄越大, 收入越高. 收入和是否为学生也会存在关系, 学生和信息评级也有关系. 为什么各个收入字段之间不独立, 还要做这种独立性假设呢, 原因就是要简化计算. 朴素贝叶斯也可以称为简单贝叶斯, 因为他做了一个把问题大大简化的假设.

### 独立性假设

在朴素贝叶斯中有两个假设

1. Two assumptions: Attributes are

• equally important a priori 字段和字段之间是同等重要的

• statistically independent (given the class value) 统计上独立性的假设,

i.e., knowing the value of one attibute says nothing about the value of another (if the class is known) 也就是说, 知道了一个字段的取值不会影响到另一个字段的取值

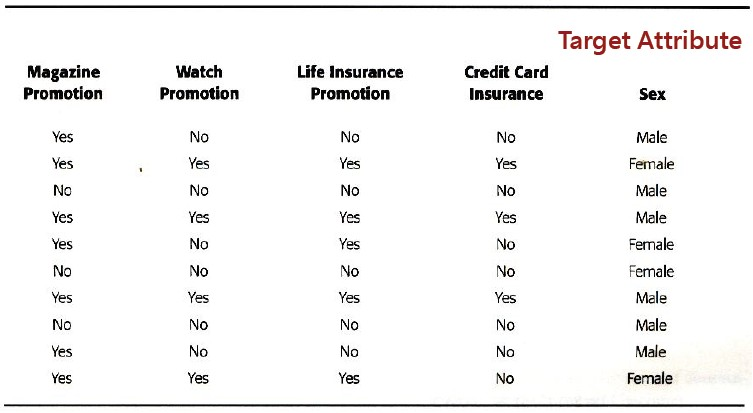
有些模型认为字段与字段之间不是同等重要的, 不同字段有不同的权重, 如逻辑斯回归. 由于不同的模型有不同的假设, 才会产生分类预测不同的结果.

2. Independence assumption is never correct! 在实际中独立性的假设从来都不正确, 也就是字段与字段之间经常不是独立的, 相互之间是有关系的.

But … often works well in practice 但是在实际操作中, 朴素贝叶斯往往能预测的很好. 原因1是朴素贝叶斯只能做分类型预测, 不能进行数值预测. 在估计yes的时候时使用独立性进行预测, 在估计no的时候也使用独立性预测, 所以yes和no之间的关系就会在一定程度上相互抵消. 原因2是, 在进行分类型预测时, 只要正确的概率大于不正确的概率即可. 即只要使用朴素贝叶斯得到的yes的概率大于得到no的概率, 就可以预测结果为yes了. 并不要求概率一定准确到小数点后多少位.

Thomas Bayes, British mathematician, 1702-1761

### 性别判断案例

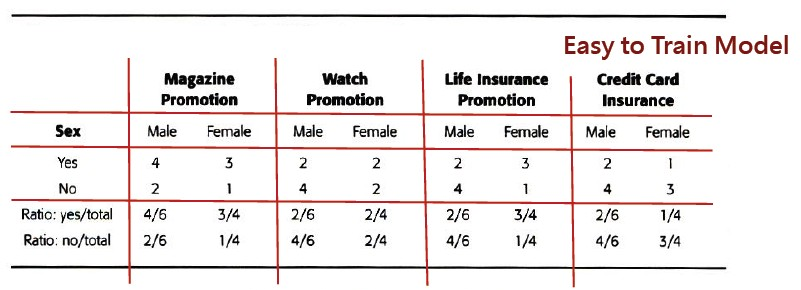


目标字段为性别, 即通过行为预测一个人是男还是女. 推荐男女在现在的大数据分析中非常重要, 如广告推荐系统, 如果知道了性别, 就可以有针对性的对不同的性别进行推荐, 这样推荐成功的概率才可能会比较大. 在有的网站中有设置性别的选项, 但对于未设置性别的用户, 就要从他的行为去预测他的性别.

在本例中, 使用4个输入字段进行预测. 即有没有接受过杂志的促销, 手表的促销, 保险的促销, 以及信用卡保险.

#### Counts and Probabilities

使用拿到的数据对朴素贝叶斯模型进行训练. 读取原数据, 计算得到下面的概率表. 相当于是对不同性别进行了分类汇总求出来的概率.



因为每个字段都是有或没有, 就可以把它们写在一张表中, 并计算产生频率值. 如果每个字段的取值并不是Yes或No, 就要对每个字段分别画出来一张表.

男生接受/不接受杂志促销的概率分别为4/6, 2/6.

女生接受/不接受杂志促销的概率分别是3/4, 1/4

从这个表中可以得到计算朴素贝叶斯所需要的所有数据. 也就是读取完一次原始的数据集之后, 就可以得到计算所需的所有数据.

也就是说, 在计算朴素贝叶斯的时候, 不是到对一个新的对象进行概率预估时才会开始计算概率, 实际上它有一个模型训练的过程, 过程就是读入原始的训练数据之后, 就会产生类似上面的这张概率表, 这张概率表就是朴素贝叶斯的模型.

所以使用朴素贝叶斯的过程非常简单, 只要读取过一次数据, 模型就产生出来了. 只要产生了这张表, 原始数据就没有用了, 只需要保留这张表就可以了.

Testing Data:

Magazine Promotion = Yes

Watch Promotion = Yes

Life insurance Promotion = No

Credit Card Insurance = No

Sex = ?

假如有一个测试数据要进行测试, 4个字段的值分别为yes, yes, no, no, 使用朴素贝叶斯来计算他的性别是男的概率高还是女的概率高.

#### Calculate for sex=male

计算在给出来的4个条件下性别为男的概率

查表可得以下几种概率:

性别为男的情况下接受magazine促销的概率

性别为男的情况下接受watch促销的概率

性别为男的情况下不接受人寿保险促销的概率

性别为男的情况下不接受信用卡保险促销的概率

性别为男的情况下满足以上4个条件的概率

所有人中性别为男的概率

满足以上4种条件的情况下性别为男性的概率为

#### Calculate for sex=female

计算在给出来的4个条件下性别为女的概率

查表可得以下几种概率:

性别为女的情况下接受magazine促销的概率

性别为女的情况下接受watch促销的概率

性别为女的情况下不接受人寿保险促销的概率

性别为女的情况下不接受信用卡保险促销的概率

性别为女的情况下满足以上4个条件的概率

所有人中性别为女的概率

满足以上4种条件的情况下性别为女性的概率为

使用下式就能计算出P(E)的值

由于性别为男的概率大于女的概率, 所以预测这个人是男性. 实际的概率使用上试计算出P(E)的值, 并把P(E)的值代入男女的概率公式中.

#### 概率为0的处理

在建立频率表时, 经常会发现很多情况下的概率值都为0, 此时不管其它的概率是多少, 只要与等于0的概率相乘, 得到的结果肯定为0. 就可能会出现预测性别为 "男" 的概率和性别为 "女" 的概率都为0. 此时就无法判断性别是 "男" 还是 "女". 所以在出现概率为0时, 要进行特殊的处理.

在使用朴素贝叶斯计算得到概率表时, 可以认为每一个格的对应的概率不是从0开始计算的, 而是从 0.5 或 1 开始的, 也就是把概率表中所有的概率都加上0.5或1. 即使概率为0, 至少还有0.5或1. 此时就能避免得到概率为0的问题. 计算分类型字段的概率时就以加 0.5 或 1 后的结果来进行计算, 这样得到的概率才是正确的.

*k* is a value betweeen 0 and 1 (usually 1)

*p* is chosen as an equal fractional part of the total number of possible values for the attribute. If an attribute has two possible values, p will be 0.5.

Let's use this technique to recompute the conditional probablility *P*(*E* | sex = female) for our previous example. With *k* = 1 and *p* = 0.5, the conditional probability of the evidence given sex = female computes to:

### 疾病诊断案例

#### Disease Diagnosis(Test Dataset)

测试数据集

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 喉咙痛 | 发烧 | 淋巴腺肿胀 | 充血 | 头痛 |  |
| Patient ID# | Sore Throat | Fever | Swollen Glands | Congestion | Headache | Diagnosis |
| 11 | No | No | Yes | Yes | Yes | ? |
| 12 | Yes | Yes | No | No | Yes | ? |
| 13 | No | No | No | No | Yes | ? |

If you are a doctor, then does the first patient get what kind of disease based on

the above 5 symptoms (Strep throat (链球菌喉炎), Allergy (过敏) or Cold (感冒))?

#### Disease Diagnosis (Training Dataset)

训练数据集

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | Target Attribute |
| Patient ID# | Sore Throat | Fever | Swollen Glands | Congestion | Headache | Diagnosis |
| 1 | Yes | Yes | Yes | Yes | Yes | Strep throat |
| 2 | No | No | No | Yes | Yes | Allergy |
| 3 | Yes | Yes | No | Yes | No | Cold |
| 4 | Yes | No | Yes | No | No | Strep throat |
| 5 | No | Yes | No | Yes | No | Cold |
| 6 | No | No | No | Yes | No | Allergy |
| 7 | No | No | Yes | No | No | Strep throat |
| 8 | Yes | No | No | Yes | Yes | Allergy |
| 9 | No | Yes | No | Yes | Yes | Cold |
| 10 | Yes | Yes | No | Yes | Yes | Cold |

#### Naive Bayes Classification

##### 计算概率值

假设有一个测试数据如下, 使用朴素贝叶斯进行分类预测

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Patient ID# | Sore Throat | Fever | Swollen Glands | Congestion | Headache | Diagnosis |
| 11 | No | No | Yes | Yes | Yes | ? |

• In Naive Bayes Classification, for the Patient 11, we need to calculate the following three probabilities:

即分别要分别计算在以上的 5 个条件的情况下得喉炎的概率, 过敏的概率和感冒的概率. 这三个概率结果哪个大, 就预测得的就是这种病.

• P(Strep Throat | Patient 11)

• P(Allergy | Patient 11)

• P(Cold | Patient 11)

基于贝叶斯公式, 能得到如下的计算公式

• Based on the Bayes Theorem, we can derive the following formulas

由于这三者都被 P(Patient 11) 相除, 所以可以把分母忽略掉

• Because they are all divided by P(Patient 11), we can omit this term.

最终的公式为

• The final formulas are listed below

If P(Strep Throat | Patient 11) > P(Allergy | Patient 11) > P(Cold | Patient 11), then we predict that the Patient 11 get the disease of Step Throat

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Patient ID# | Sore Throat | Fever | Swollen Glands | Congestion | Headache | Diagnosis |
| 1 | Yes | Yes | Yes | Yes | Yes | Strep throat |
| 2 | No | No | No | Yes | Yes | Allergy |
| 3 | Yes | Yes | No | Yes | No | Cold |
| 4 | Yes | No | Yes | No | No | Strep throat |
| 5 | No | Yes | No | Yes | No | Cold |
| 6 | No | No | No | Yes | No | Allergy |
| 7 | No | No | Yes | No | No | Strep throat |
| 8 | Yes | No | No | Yes | Yes | Allergy |
| 9 | No | Yes | No | Yes | Yes | Cold |
| 10 | Yes | Yes | No | Yes | Yes | Cold |

从表中可以计算出来

现在关键是要计算在诊断结果是 Strep Throat 的情况下是 Patient 11的概率, 也就是计算在诊断结果是 Strep Throat 的情况下是 Patient 11 的5个特征的概率.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Patient ID# | Sore Throat | Fever | Swollen Glands | Congestion | Headache | Diagnosis |
| 11 | No | No | Yes | Yes | Yes | ? |

在诊断结果是 Strep Throat 的情况下有5个特征的概率等于分别有 5 个分立特征的概率的乘积, 即

由于

所以

用同样的方法就可以计算在 Patient 11 的情况下得 Allergy 的概率和得 Cold 的概率

Please Calculate the following two probabilities

##### 概率正规化

• Adjust the probabilities to satisfy the probability theorem

• Before

• P(Strep Throat | Patient 11) = 1 / (5\*3\*3\*3)=0.0074

• P(Allergy | Patient 11) = 0

• P(Cold | Patient 11) = 0

正规化就是把上面三者之和做分母, 去除原来的值.

• After

• P(Strep Throat | Patient 11) = 0.0074 / (0.0074 + 0 + 0) = 1.0

• P(Allergy | Patient 11) = 0 / (0.0074 + 0 + 0) = 0.0

• P(Cold | Patient 11) = 0 / (0.0074 + 0 + 0) = 0.0

正规化之后得到的结果是 Patient 11得 Strep Throat的概率是100%. 在有些情况下甚至会得到所有可能的概率都为 0, 就需要对 0 值进行处理.

##### 输入字段空值的处理

在测试数据存在空值的条件下, 选择视而不见.

在朴素贝叶斯中, 有一个很好的特性, 即对于空值, 并不需要事先进行填补. 如下面4个字段中, Watch Promotion = Unknown, 就相当于是空值.

Magazine Promotion = Yes

Watch Promotion = Unknown

Life Insurance Promotion = No

Credit Card Insurance = No

Sex = ?

在朴素贝叶斯中就对空值视而不见, 相当于没有那个条件或字段, 使用余下的3个条件来计算概率. 分别计算在 "性别为男" 的前提下出现以上3种条件的概率和 "性别为女" 的条件下出现以上3种条件的概率

训练数据集中出现空值时也是选择 "视而不见" 的方法, 在计算概率计数时直接路过空值即可.

### 数值型字段的处理

#### 数值型字段处理的原则

以上情况下都是处理的类别型的字段, 类别型的字段才比较好计算概率. 但得到的数据中很多都是数值型的字段, 要事先进行处理.

• 朴素贝叶斯在数值型字段的处理，方法有二

• 将数值型属性离散化，使数值型属性变成类别型属性. 比如说年龄, 把它切割成几个区间, 如切割成 6 个年龄段, 这样就可以当成类别型属性来处理.

• 以正态分布的公式，直接处理数值型属性. 在假设数值型字段服从正态分布的前提下, 直接使用正态分布的公式处理数值型的字段.

#### 正态分布处理

正态分布公式

where

*e* = the exponential function

μ = the class mean for the given numerical attribute 均值

σ = the class standard deviation for the attribute 标准差

*x* = the attribute value

New Data: 多了一个年龄的字段.

Magazine Promotion = Yes

Watch Promotion = Yes

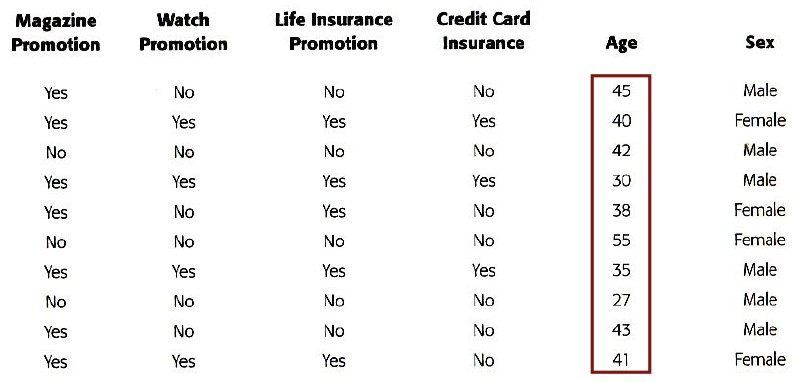
Life Insurance Promotion = No

Credit Card Insurance = No

Age = 45

Sex = ?

#### 多出来年龄字段时性别的预测



#### 使用正态分布公式计算概率

计算在性别分别为男和女的情况下满足以上5种情况的概率.

现在的问题就是要求在性别分别为男和女的情况下年龄为45岁的概率.

首先要计算性别为男的情况下的均值和标准差. 然后通过正态分布公式来计算在性别为男的前提下年龄为45的概率.

In Table to find the mean and standard deviation scores. For the class sex = male, we have: μ = 37.00, σ = 7.69, and x = 45. Therefore the probability that age = 45 given sex = male is computed as:

Making the computation, we have:

**计算女性的概率**

To determine the conditional probability for age given sex = female, we substitute μ = 43.50, σ = 7.77, and x = 45. Specifically,

Making the computation, we have

We can now determine the overall conditional probability values:

# 使用正态分布公式计算概率值

# 对讲义上年龄为45的数值型进行预测, 看对应不同性别的概率. 男性均值为37, 女性均值为43.5

dnorm(45, mean=37.0, sd=7.69)

# [1] 0.03019787

dnorm(45, mean=43.5, sd=7.77)

# [1] 0.05039603

Once again, we ignore P(E) and conclude that the instance belongs to the male class.

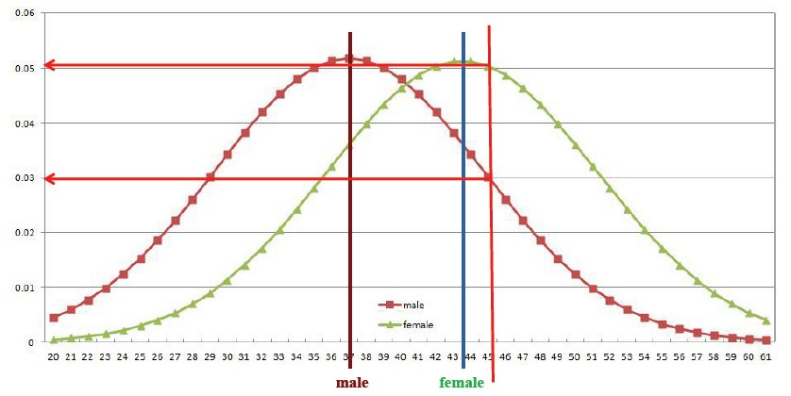
对以上的结果再次进行一次正规化, 就得到了实际的概率值.

#### 从正态分布图上计算概率

已知知道了均值和标准差, 假设男女的数据都服从正态分布, 就可以画出以下的男女正态分布图.

然后有一个测试数据, 年龄为45岁, 看这个值是更接近于性别为男的均值还是更接近于性别为女的均值. 与哪个均值越近, 它是对应性别的概率就越大, 反之越远.

根据正态分布的公式就可以计算出年龄为45对应于男和女的概率. 从图上也可以看出, 年龄为45时对应女性的概率为0.05, 对应男性的概率为0.03.



#### 朴素贝叶斯的优点

朴素贝叶斯分类法有非常多好的特性，尤其在Big Data的环境中.

同时，它支持Incremental Learning/Updatable Learning，渐进式学习/修正式的学习, 非常难得.

什么是渐进式/修正式的学习呢, 原训练数据集有10个数据, 计算得到一个概率表/频率表. 在原数据又增加10条数据的情况下, 就可以在原概率表的基础上继续累加, 更新频率, 就得到一个新的模型. 每一笔新的数据进来就更新一次, 每次都可以得到最新的模型. 这就是updatable Learning的思想. 即在旧的模型上修正更新, 就会得到新模型.

但是几乎没有一个数据挖掘的模型是渐进式学习/修正式的模型, 如决策树和神经网络, 都不支持模型更新修改. 在数据量为10时, 建立起一个模型, 在数据量增加在20时, 必须重新使用所有的20笔数据再次建模, 才能得到新的模型, 不能在原来模型的基础上更新. 当数据量变得越来越大时, 这些模型的实现就越来越困难. 或者会使用抽样的方式建立模型.

朴素贝叶斯因为支持update模型, 所以是一个非常难得的模型. 实际中也有很多情况都使用朴素贝叶斯进行预测, 并且得到的结果往往会比其它模型都好, 如判断邮件是否是垃圾邮件. 病毒判断.

## 贝式网络 BAYES NET

### TAN模型

贝式网络中的TAN模型与朴素贝叶斯的不同之处在于朴素贝叶斯是建立在输入字段之间是独立的这个假设上, 做这个简化的目就是简化和加速计算. TAN模型也是在朴素贝叶斯基础上的一个扩充, 放宽独立性的假设, 即不再要求字段之间是独立的, 可以对存在相互关系的输入字段进行处理.

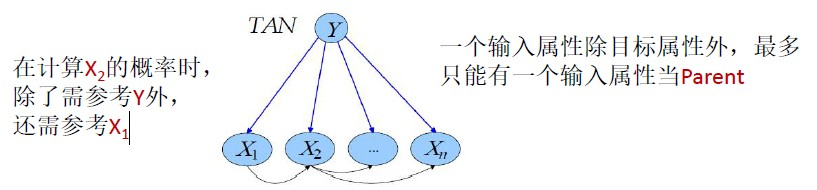
• Tree Augmented Naive Bayesian (TAN) is used to relax independence

assumption

• It is defined by the following conditions

• Target attribute Y is the parent of all input attributes

• One Input attribute may have one other input attribute as the parent



在上面这张图中, 目标字段为 Y, 因为 Y 是通过贝氏定理计算得到的, 即计算各个 X 已知的条件下 Y 的概率, 反过来, 计算 Y 已知的条件下各个 X 的概率, 原来朴素贝叶斯中的 Y 会影响各个 X 的概率的计算, 此时Y 就变成了条件, 成为了输入变量, X 就成了目标变量. 在TAN模型中, 这个条件放宽了, 一个输入属性除目标属性外, 最多只能有一个输入属性当 Parent. 以X2为例来说明, 在从Y的概率计算X2的概率时, 只有Y会影响X2的概率, 但是从图中可以看到, X1也会影响X2的概率, 即X2的影响条件除了Y之外又多出来了一个X1, TAN模型规定, 一个输入字段X除了目标字段Y之外, 最多只能有一个输入字段X1当Parent, 即X2会被Y和X1影响. 这里所说的是最多, 所以也可以没有, 即不是所有的输入字段都会受到一个其它输入字段的影响. 所以在计算X2的概率时, 除了需要参考Y之外, 还要参考X1的值.

TAN字段的核心就是放宽了朴素贝叶斯中关于输入字段之间独立性的假设, 但也不能放的太宽了, 因为如果有很多字段都会影响到X2的话, 计数为 0 的部分就会越来越多, 数据量就会变得非常大. 假如Y有Yes和No两种情况, X2有三种情况, 就要计算二者组合的6种概率, 如果X1也有三种取值, 再把X1计算进去, 三者组合就要计算18种概率, 这18种概率中很多都是0值, 这样就没法计算了. 这就是TAN模型中规定一个输入属性最多只能受另外一个输入属性影响的原因.

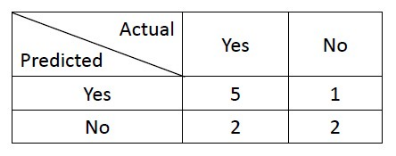
## 模型评估

到目前为止都是使用正确率来进行评估的, 在之后的使用中, 根据不同的应用场景, 评估的方式可能会发生变化.

如下有10个客户的数据, 对模型评估的结果和标准答案进行对比.

|  |  |  |
| --- | --- | --- |
| Customer ID | Model Predicted Result | Actual Result |
| 1 | Yes | Yes |
| 2 | No | Yes |
| 3 | Yes | Yes |
| 4 | No | No |
| 5 | Yes | No |
| 6 | Yes | Yes |
| 7 | No | No |
| 8 | No | Yes |
| 9 | Yes | Yes |
| 10 | Yes | Yes |

### 模型评估-Confusion Matrix



• Accuracy 正确率

• Precision 回应率、命中率

• Recall 捕捉率、查全率

• F-measure F指标，同时考虑Precision & Recall

### 正确率

实际Yes, 预测Yes, 实际No, 预测No.

### 回应率, 命中率

如在营销活动中, 预测为Yes的情况下就会对客户进行营销, 在所有进行营销的客户中, 客户接受的比率有多大. 预测为Yes的有6个人, 其中5个人会接受营销, 所以回应率为5/6. 在营销活动中, 经常使用命中率, 不使用正确率, 如营销活动中平均100个人只有1个人能接受, 10000个人中只有100个人能接受, 如果模型预测所有人对这个产品都没有兴趣, 因为接受率是1%, 全部预测不接受, 那正确率就是99%, 但这样高的正确率对公司的营销活动不起到任何的作用, 所以公司一般都比较关注回应率, 即预估可能会购买公司产品或服务的的人中实际会购买的概率.

回应率有时候也可以做假, 即通过一定的方法提高回应率. 在预估结果为Yes或No时不仅要得出来Yes或No的结果, 还会得到 Yes 或 No 对应的概率, 一般情况下, 得到的Yes的概率高于No的概率, 就认为它是Yes, 即经过标准化后Yes的概率大于50%, 就认为结果是Yes. 但是可以把这个标准定的更严格一些, 如认为Yes的概率必须要大于90%才能认为结果为Yes, 假如Yes的概率定为大于50%时得到的Yes的人数为100人, 而Yes的概率为大于90%时得到的Yes的人数为10人. 一般来说, 概率越高, 模型得到的结果与实际相符合的可能性就越高, 设置概率为90%, 得到了10个客户, 对这10个客户进行营销, 命中率可能为100%, 即10个客户全部都接受了推销. 但一个营销案例才对10个人进行营销, 这也不是公司希望看到的结果. 所以此时还要关注捕捉率.

### 捕捉率

捕捉率主要是看有没有漏网之鱼,模型预测得到的人数与全部会来买的人数之比, 就是捕捉率. 在这个案例中, 捕捉到5个人, 一共有7个人会来买产品, 捕捉率为5/7.

在上面使用高概率预测得到的高捕捉率的情况下, 捕捉率可能会很低. 因为预测得到了10个数据, 而实际可能来购买的人数可能为1000人.

捕捉率也可以做假, 如修改模型条件, 把所有人都认为是Yes, 对所有的客户进行营销, 这样得到的捕捉率就是100%, 即所有来购买的客户都是我们模型中预测会来购买的人. 但此时的回应率可能就会很低了. 可能为了获取到这1000个来购买的人, 而去对100W人进行了营销, 此时的回应率就很低.

所以在数据竞赛中, 一般都不看正确率, 而是关注回应率和捕捉率.

### F-指标

F-指标就同时考虑了回应率和捕捉率. 即2倍的回应率乘以捕捉率除以回应率与捕捉率之和. F-指标一般都在0到1之间, 越靠近1, 模型预测的效果就越好.

F-指标的特性.

假如有3个模型, 对同一个数据集进行预测, 得到的回应率和捕捉率如下

|  |  |  |  |
| --- | --- | --- | --- |
| P | R | 解释 | F |
| 0.1 | 0.9 | 捕捉率很高, 漏网之鱼很少 | 0.18 |
| 0.9 | 0.1 | 命中率很高, 漏网之鱼很多 | 0.18 |
| 0.5 | 0.5 |  | 0.5 |

即如果回应率和捕捉率不一样, F值就会倾向于二者中较低的一个, 比较低的那个值稍微高一点.

如果回应率和捕捉率相同, F值就与它们相同.

F指标就可以避免过度强调回应率或过度强调捕捉率, 在二者之间找到了一个平衡.

## R语言实作

• 以R语言进行朴素贝叶斯模型建置及算法参数调整

# Read Diseases Dataset

# 读取数据集

data <- read.csv("D:/R\_edu/data/diseases.csv")

# 把patient\_id字段去除掉, 得到的前面5个是输入字段, 后面1个是目标字段, 并且都是类别型字段

data <- data[,-1]

head(data)

# Patient\_ID Sore\_Throat Fever Swollen\_Glands Congestion Headache Diagnosis

# 1 1 Yes Yes Yes Yes Yes Strep\_throat

# 2 2 No No No Yes Yes Allergy

# 3 3 Yes Yes No Yes No Cold

# 4 4 Yes No Yes No No Strep\_throat

# 5 5 No Yes No Yes No Cold

# 6 6 No No No Yes No Allergy

# 产生训练集和测试集

# Generate Training & Test Datasets

set.seed(102)

select <- sample(1:nrow(data),nrow(data)\*0.8)

train <- data[select,]

test <- data[-select,]

# Build Naive Bayes Model with No Laplace Smoothing

# 建立朴素贝叶斯模型

# Laplace Smoothing就是指要不要进行空值的处理. 预测是不做空值的处理的.

library(e1071)

# naiveBayes函数来进行朴素贝叶斯预测. 目标字段为Diagnosis, 一个~加一个句号表示除了目标字段之外的所有字段都作为输入字段来处理. data就是训练数据集train

# 先生成一个朴素贝叶斯的模型

nb\_default <- naiveBayes(Diagnosis ~ ., data=train)

# 使用nb\_default这个朴素贝叶斯的模型对测试数据集test进行预测, 要对测试数据test进行预测, type="class"会输出预测测试数据的类别.

test.y\_hat <- predict(nb\_default, test, type="class")

test.y\_hat

# 测试数据只有2笔资料, 一个预测结果为Allergy, 一个为Cold

# [1] Allergy Cold

# Levels: Allergy Cold Strep\_throat

# 使用type="raw"来输出预测的概率值

predict(nb\_default, test, type="raw")

# 第1笔资料预测为Allergy的概率为0.9, 第二笔资料预测为Cold的概率为0.99, Strep\_throat为二者之外的概率

# Allergy Cold Strep\_throat

# [1,] 0.907299343 0.06641788 2.628277e-02

# [2,] 0.001984737 0.99796829 4.697541e-05

# 取概率值的小数点后3位

test.y\_hat\_prob <- round(predict(nb\_default, test, type="raw"),3)

# 得到的概率值会自动正规化为100%

test.y\_hat\_prob

# Allergy Cold Strep\_throat

# [1,] 0.907 0.066 0.026

# [2,] 0.002 0.998 0.000

# 把预测的结果和对应的概率值通过列合并合并到一起

cbind(Prediction=as.character(test.y\_hat), test.y\_hat\_prob)

# Prediction Allergy Cold Strep\_throat

# [1,] "Allergy" "0.907" "0.066" "0.026"

# [2,] "Cold" "0.002" "0.998" "0"

# 对预测结果的准确度进行评估. test.y\_hat为预测结果, test$Diagnosis为标准答案.

# Model Evaluation

accuracy.nb\_default <- sum(test.y\_hat==test$Diagnosis) / length(test$Diagnosis)

# 两个值都一样, 准确率为100%

accuracy.nb\_default

# [1] 1

# 看两个预测结果是否一致

agreement\_KNN <- test.y\_hat==test$Diagnosis

agreement\_KNN

# [1] TRUE TRUE

# 列联表分析, 并给列联表的字段自定义名字

table(test.y\_hat, test$Diagnosis, dnn=c("Prediction","Actual"))

# Actual

# Prediction Allergy Cold Strep\_throat

# Allergy 1 0 0

# Cold 0 1 0

# Strep\_throat 0 0 0

# 数据都保存在nb\_default中

nb\_default

#

# Naive Bayes Classifier for Discrete Predictors

#

# Call:

# naiveBayes.default(x = X, y = Y, laplace = laplace)

#

# A-priori probabilities:

# Y

# 在没有任何附加条件的情况下, 三者出现的概率值

# Allergy Cold Strep\_throat

# 0.250 0.375 0.375

#

# Conditional probabilities:

# Patient\_ID

# Y [,1] [,2]

# Allergy 7 1.414214

# Cold 6 3.605551

# Strep\_throat 4 3.000000

#

# Sore\_Throat

# Y No Yes

# Allergy 0.5000000 0.5000000

# Cold 0.3333333 0.6666667

# Strep\_throat 0.3333333 0.6666667

#

# Fever

# Y No Yes

# Allergy 1.0000000 0.0000000

# Cold 0.0000000 1.0000000

# Strep\_throat 0.6666667 0.3333333

#

# 对应的Swollen\_Glands的次数

# Swollen\_Glands

# Y No Yes

# Allergy 1 0

# Cold 1 0

# Strep\_throat 0 1

#

# Congestion

# 对应三种诊断结果的情况下是否有Congestion这个症状的概率. No表示没有Congestion这个症状, Yes表示有这个症状. 即过敏的情况下100%充血, 感冒的情况下100%充血, 喉炎的情况下有2/3充血, 1/3不充血

# Y No Yes

# Allergy 0.0000000 1.0000000

# Cold 0.0000000 1.0000000

# Strep\_throat 0.6666667 0.3333333

#

# Headache

# Y No Yes

# Allergy 0.5000000 0.5000000

# Cold 0.6666667 0.3333333

# Strep\_throat 0.6666667 0.3333333

# 查看某一种症状对应的三种诊断结果的概率

nb\_default$tables$Sore\_Throat

# Sore\_Throat

# Y No Yes

# Allergy 0.5000000 0.5000000

# Cold 0.3333333 0.6666667

# Strep\_throat 0.3333333 0.6666667

# Build Naive Bayes Model with Laplace Smoothing = 1

library(e1071)

# 要进行空值的填补, Laplace Smoothing =1 表示预测从1开始计数, 相当于所有数据都加上了1, 就避免了结果概率为0的问题.

nb\_laplace1 <- naiveBayes(Diagnosis ~ ., data=train, laplace=1)

test.y\_hat <- predict(nb\_laplace1, test, type="class")

test.y\_hat\_prob <- round(predict(nb\_laplace1, test, type="raw"),3)

cbind(Prediction=as.character(test.y\_hat), test.y\_hat\_prob)

# Prediction Allergy Cold Strep\_throat

# [1,] "Allergy" "0.714" "0.208" "0.078"

# [2,] "Cold" "0.212" "0.742" "0.046"

# Model Evaluation

accuracy.nb\_default <- sum(test.y\_hat==test$Diagnosis) / length(test$Diagnosis)

accuracy.nb\_default

# [1] 1

agreement\_KNN <- test.y\_hat==test$Diagnosis

agreement\_KNN

# [1] TRUE TRUE

table(test.y\_hat, test$Diagnosis, dnn=c("Prediction","Actual"))

# Actual

# Prediction Allergy Cold Strep\_throat

# Allergy 1 0 0

# Cold 0 1 0

# Strep\_throat 0 0 0

# 对空值进行填补, 数据都加1的情况, 得到的结果会稍微不同

nb\_laplace1$tables$Sore\_Throat

# Sore\_Throat

# Y No Yes

# Allergy 0.5 0.5

# Cold 0.4 0.6 分子原来分别为1,2, 加1变为2,3, 二者相加为5, 概率分别为2/5, 3/5.

# Strep\_throat 0.4 0.6

# 未进行空值填补的情况下的概率

nb\_default$tables$Sore\_Throat

# Sore\_Throat

# Y No Yes

# Allergy 0.5000000 0.5000000

# Cold 0.3333333 0.6666667

# Strep\_throat 0.3333333 0.6666667

# 输入字段为数值型字段的处理. 预测老板今天会不会去打高尔夫球. 可能与天气, 温度, 温度和有没有风有关系. 输入字段中有两个类别型两个数值型的字段

# Read Weather Dataset

data <- read.csv("D:/R\_edu/data/Weather-Numeric.csv")

head(data)

# outlook temperature humidity windy play

# 1 sunny 85 85 FALSE no

# 2 sunny 80 90 TRUE no

# 3 overcast 83 86 FALSE yes

# 4 rainy 70 96 FALSE yes

# 5 rainy 68 80 FALSE yes

# 6 rainy 65 70 TRUE no

# Generate Training & Test Datasets

set.seed(102)

select <- sample(1:nrow(data),nrow(data)\*0.8)

train <- data[select,]

test <- data[-select,]

# Build NB Model with No Laplace Smoothing

library(e1071)

# 朴素贝叶斯使用正态分布的公式进行数值型字段概率的推估, 所以不用进行特殊的设置

nb\_default <- naiveBayes(play ~ ., data=train)

nb\_default

# Naive Bayes Classifier for Discrete Predictors

#

# Call:

# naiveBayes.default(x = X, y = Y, laplace = laplace)

#

# A-priori probabilities:

# Y

# no yes

# 0.3636364 0.6363636

#

# Conditional probabilities:

# outlook

# outlook有三个值,得到这三个值对应的概率. 目标字段Y有yes和no, 由于没有进行概率为0的处理, 会得到有些概率为0的情况.

# Y overcast rainy sunny

# no 0.0000000 0.5000000 0.5000000

# yes 0.4285714 0.2857143 0.2857143

#

# temperature

# 数值型的字段, 使用平均值和标准差

# Y [,1] [,2]

# no 73.25000 8.421203

# yes 73.85714 6.743604

#

# humidity

# Y [,1] [,2]

# no 85.25000 10.96586

# yes 77.42857 10.76812

#

# windy

# 类别型的字段, 有风无风.

# Y FALSE TRUE

# no 0.5000000 0.5000000

# yes 0.7142857 0.2857143

test.y\_hat <- predict(nb\_default, test, type="class")

test.y\_hat

# [1] no yes yes

# Levels: no yes

test.y\_hat\_prob <- round(predict(nb\_default, test, type="raw"),3)

test.y\_hat\_prob

# 得到训练数据集中三个记录对应是否去打高尔夫球的概率

# no yes

# [1,] 0.731 0.269

# [2,] 0.377 0.623

# [3,] 0.003 0.997

# 把结果和概率拼接到一起

cbind(Prediction=as.character(test.y\_hat), test.y\_hat\_prob)

# Prediction no yes

# [1,] "no" "0.731" "0.269"

# [2,] "yes" "0.377" "0.623"

# [3,] "yes" "0.003" "0.997"

# Model Evaluation

accuracy.nb\_default <- sum(test.y\_hat==test$play) / length(test$play)

accuracy.nb\_default

# 预测的准确度为100%

# [1] 1

agreement\_KNN <- test.y\_hat==test$play

agreement\_KNN

# [1] TRUE TRUE TRUE

table(test.y\_hat, test$play, dnn=c("Prediction","Actual"))

# Actual

# Prediction no yes

# no 1 0

# yes 0 2

# 查看湿度的情况下是否会去打球. 会去打球时湿度是比较低的.

nb\_default$tables$humidity

# humidity

# Y [,1] [,2]

# no 85.25000 10.96586

# yes 77.42857 10.76812

# 查看天气的情况下是否会去打球, 阴天一般最喜欢去打球, 而不去打球时不是下雨就是晴天, 这也符合生活中的常识

nb\_default$tables$outlook

# outlook

# Y overcast rainy sunny

# no 0.0000000 0.5000000 0.5000000

# yes 0.4285714 0.2857143 0.2857143

# 使用正态分布公式计算概率值

# 对讲义上年龄为45的数值型进行预测, 看对应不同性别的概率. 男性均值为37, 女性均值为43.5

dnorm(45, mean=37.0, sd=7.69)

# [1] 0.03019787

dnorm(45, mean=43.5, sd=7.77)

# [1] 0.05039603

# 输入字段中存在着数值型字段的情况下对概率为0的情况进行处理

# Build KNN Model with Laplace Smoothing = 1

library(e1071)

nb\_laplace1 <- naiveBayes(play ~ ., data=train, laplace=1)

test.y\_hat <- predict(nb\_laplace1, test, type="class")

test.y\_hat\_prob <- round(predict(nb\_laplace1, test, type="raw"),3)

cbind(Prediction=as.character(test.y\_hat), test.y\_hat\_prob)

# Prediction no yes

# [1,] "no" "0.689" "0.311"

# [2,] "yes" "0.382" "0.618"

# [3,] "yes" "0.342" "0.658"

# Model Evaluation

accuracy.nb\_default <- sum(test.y\_hat==test$play) / length(test$play)

accuracy.nb\_default

# [1] 1

agreement\_KNN <- test.y\_hat==test$play

agreement\_KNN

# [1] TRUE TRUE TRUE

table(test.y\_hat, test$play, dnn=c("Prediction","Actual"))

# Actual

# Prediction no yes

# no 1 0

# yes 0 2

nb\_laplace1$tables$humidity

# humidity

# Y [,1] [,2]

# no 85.25000 10.96586

# yes 77.42857 10.76812

# 对概率为0的情况进行处理的结果

nb\_laplace1$tables$outlook

# outlook

# Y overcast rainy sunny

# no 0.1428571 0.4285714 0.4285714

# yes 0.4000000 0.3000000 0.3000000

# 不对概率为0的情况进行处理的结果

nb\_default$tables$outlook

# outlook

# Y overcast rainy sunny

# no 0.0000000 0.5000000 0.5000000

# yes 0.4285714 0.2857143 0.2857143

# 朴素贝叶斯对第12章KNN的作业进行模拟

# Read Broadband Dataset

data <- read.csv("D:/R\_edu/data/broadband.csv")[,-1]

# Gender, Channel, Autopay, Broadband是类别型的, 要把类别型的字段转换为factor

for(i in c(1,4,5,12)) {

data[[i]] <- as.factor(data[[i]])

}

# Generate Training & Test Datasets

set.seed(102)

select <- sample(1:nrow(data),nrow(data)\*0.8)

train <- data[select,]

test <- data[-select,]

# Build Naive Bayes Model

library(e1071)

# 对概率为0进行处理, 从1开始计数

nb\_laplace1 <- naiveBayes(BROADBAND ~ ., data=train, laplace=1)

test.y\_hat <- predict(nb\_laplace1, test, type="class")

# 因为round为3位小数, 所以得到的结果中还有很多0值. 如果取10位小数, 得到结果为0的概率就会少很多.

test.y\_hat\_prob <- round(predict(nb\_laplace1, test, type="raw"),3)

cbind(Prediction=as.character(test.y\_hat), test.y\_hat\_prob)

# Model Evaluation

accuracy.nb\_default <- sum(test.y\_hat==test$BROADBAND) / length(test$BROADBAND)

# 结果为0.89,比knn的预测精度高了一点

accuracy.nb\_default

# [1] 0.8923767

agreement <- test.y\_hat==test$BROADBAND

agreement

# 交叉表

table(test.y\_hat, test$BROADBAND, dnn=c("Prediction","Actual"))

# Actual

# Prediction 0 1

# 0 180 17

# 1 7 19

# 查看数据的正态性

# Read Broadband Dataset

data <- read.csv("D:/R\_edu/data/broadband.csv")[,-1]

# 画出来不是正态分布, 右边有严重的离群值, 所以为右偏分布, 所以使用正态分布来估计, 结果可能不会太好. 把它离散化, 分割成几个区间, 也许得到的结果会更好一些. 贝叶斯就比较擅长处理类别型的字段的数据.

hist(data$ARPB\_3M, breaks=50)

skewness(data$ARPB\_3M) # Right Skewness值大于0时为右偏分布

# [1] 4.735234

# 总结, 朴素贝叶斯可以接受数值型也可以接受类别型的字段/变量, 但更擅长于处理类别型的数据. 数值型的变量是由正态分布公式直接计算出它的概率, 但是通过对数据做直方图可以看出, 它并不服从正态分布, 所以使用正态分布来计算数值型概率并不是很合适.

• 以R语言进行贝式网络模型建置及算法参数调整