Python 软件

2022-2023 秋

2022 年 11 月 30 日

朴素贝叶斯的实现

Week13

2022 年 11 月 30 日

目录

主要内容

主要内容

基本公式与独立性公式

X, Y 是两个事件

- 条件概率公式定义

$$P(X|Y) = \frac{P(X, Y)}{P(Y)}$$

- 交换 X,Y 得到

$$P(Y|X) = \frac{P(X, Y)}{P(X)}$$

- 所以

$$P(X, Y) = P(X|Y)P(Y) = P(Y|X)P(X)$$

- 于是

$$P(X|Y) = \frac{P(Y|X)P(X)}{P(Y)}$$

条件独立性

$$P(X, Y|C) = P(X|C)P(Y|C)$$

主要内容

贝叶斯与朴素贝叶斯

贝叶斯推断

目标

$$P(Y|x_1, x_2, ..x_D)$$

方法

$$P(Y|x_1, x_2, ..x_D) = \frac{P(x_1, x_2, ..x_D|Y)P(Y)}{\sum_Y P(x_1, x_2, ..x_D|Y)P(Y)}$$

一般贝叶斯

- **完全联合概率计算**

$$P(x_1, x_2, ..x_D|Y)$$

朴素贝叶斯

- 对数据做条件独立性假设, 假设给定 Y 时, $x_1, x_2, ..x_D$ 彼此独立

$$P(x_1, x_2, ..x_D|Y) = \prod_{d=1}^{D} P(x_d|Y)$$

主要内容

主要内容

数据说明

记录在不同天气的情况下, Tom 和 Jerry 外出打网球的情况, 数据如下

	Outlook	Temp	Humi	Wind	play
0	Sunny	Hot	High	Weak	no
1	Sunny	Hot	High	Strong	no
2	Overcast	Hot	High	Weak	yes
3	Rain	Mild	High	Weak	yes
4	Rain	Cool	Normal	Weak	yes
5	Rain	Cool	Normal	Strong	no
6	Overcast	Cool	Normal	Strong	yes
7	Sunny	Mild	High	Weak	no
8	Sunny	Cool	Normal	Weak	yes
9	Rain	Mild	Normal	Weak	yes
10	Sunny	Mild	Normal	Strong	yes
11	Overcast	Mild	High	Strong	yes
12	Overcast	Hot	Normal	Weak	yes
13	Rain	Mild	High	Strong	no

- 一行对应一个样本，Outlook, Temp, Humi, Wind 为样本的属性, play为样本标签

主要内容

要回答的问题

期望基于已知数据预测不同的天气状况下, Tom 和 Jerry 外出打球的可能性

1. 在 Sunny, Cool, High, Strong 的情况下他们会不会出去打球?
2. 在 Overcast, Hot, Normal 的情况下他们会不会出去打球?
3. 在 Rain, Cool, Weak 的情况下他们会不会出去打球?

三个对应的概率值的计算

$$P(play = yes | Outlook = Sunny, Temp = Cool, Humi = High, Wind = Strong)$$
$$P(play = yes | Outlook = Overcast, Temp = Hot, Humi = Normal)$$
$$P(play = yes | Outlook = Rain, Temp = Cool, Wind = Weak)$$

问题回答 (以第一个问题为例)

- 将 Outlook = Sunny, Temp = Cool, Humi = High, Wind = Strong 这四种情况同时发生的事件记作 X
- 事件 play = yes 记作 Y
- 目标 $P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$

问题回答 (以第一个问题为例, 续)

X={Outlook=Sunny,Temp=Cool,Humi=High, Wind=Strong},Y={play=yes}

$$P(\underbrace{play = yes}_{Y} | \underbrace{Outlook = Sunny, Temp = Cool, Humi = High, Wind = Strong}_{X})$$

$$= \frac{P(\underbrace{Outlook = Sunny, Temp = Cool, Humi = High, Wind = Strong}_{X} | \underbrace{play = yes}_{Y}) P(\underbrace{play = yes}_{Y})}{P(\underbrace{Outlook = Sunny, Temp = Cool, Humi = High, Wind = Strong}_{X})}$$

$P(X)$ 的计算

- $P(X) = \sum_Y P(X|Y)P(Y)$
- $P(Outlook = Sunny, Temp = Cool, Humi = High, Wind = Strong) =$
  $P(Outlook = Sunny, Temp = Cool, Humi = High, Wind = Strong|play = yes)P(play = yes) +$
  $P(Outlook = Sunny, Temp = Cool, Humi = High, Wind = Strong|play = no)P(play = no)$
- $P(play = yes)$ 的计算是直接的, 难点在于计算
  $P(Outlook = Sunny, Temperature = Cool, Humidity = High, Wind = Strong|play)$

主要内容

朴素贝叶斯假设

朴素贝叶斯假设给定 y 的情况下, X 中各子事件条件独立

$$
\begin{aligned}
&P(Outlook = Sunny, Temperature = Cool, Humidity = High, Wind = Strong | play = yes) \\
= \ &P(Outlook = Sunny | play = yes) \times P(Temp = Cool | play = yes) \\
&\times P(Humi = High | play = yes) \times P(Wind = Strong | play = yes)
\end{aligned}
$$

各个相关计算项

- P(play = yes), P(Outlook = Sunny|play = yes), P(Temp = Cool|play = yes), P(Humi= High|play = yes), P(Wind = Strong|play = yes)
- P(play = no), P(Outlook = Sunny|play = no), P(Temp = Cool|play = no), P(Humi= High|play = no), P(Wind = Strong|play = no)

⋆ **思考: 在上述项的计算中, 变化的是什么, 不变的是什么?**

链接

主要内容

主问题分解

计算 $P(play = yes|Outlook = Sunny, Temp = Cool, Humi = High, Wind = Strong)$ 步骤

1. **计算联合概率 P_1**
   - $P_1 = P(Outlook = Sunny, Temp = Cool, Humidity = High, Wind = Strong, play = yes)$
2. **计算联合概率 P_2**
   - $P_2 = P(Outlook = Sunny, Temp = Cool, Humidity = High, Wind = Strong, play = no)$
3. **返回后验概率 $\frac{P_1}{P_1+P_2}$**

play 的取值 yes 或 no

1. **遍历标签的值的集合, 分别计算对应的联合概率**
2. **Step1,2 使用循环可完成, 并且能够适用更广泛的标签有任意类别数目的情形.**

子问题分解

与标签值相关的联合概率的计算

计算 $P = P(Outlook = Sunny, Temp = Cool, Humidity = High, Wind = Strong, play = tag)$, P_1, P_2 分别对应 tag 等于 yes 或者 no 的情况.

1. 获取当前标签值 tag,
2. 计算 $P(play = tag)$ 并作为输出概率 P 的初值.
3. 计算 $p = P(Outlook = Sunny \mid play = tag)$, 更新 $P = P * p$
4. 计算 $p = P(Temperature = Cool \mid play = tag)$, 更新 $P = P * p$
5. 计算 $p = P(Humidity = High \mid play = tag)$, 更新 $P = P * p$
6. 计算 $p = P(Wind = High \mid play = tag)$, 更新 $P = P * p$
7. 返回最终的联合概率 P

Highlights

1. 通过观察第 3 步到第 6 步可知, 变化的内容是 Outlook = Sunny, Temp= Cool, Humidiy = High, Wind = High.
2. 遍历属性名称 attr 便能获得对应的属性值 value, 计算 $P(attr = value|play = tag)$
3. 第 3 步到第 6 步便能通过循环完成, 并且能够适应输入为任意数目的属性及其值的情形.

Python 数据的准备

数据存储在某 csv 文件 Play.csv *s3f21.py*

Play.csv

```
1  Outlook,Temp,Humi,Wind,play
2  Sunny,Hot,High,Weak,no
3  Sunny,Hot,High,Strong,no
4  Overcast,Hot,High,Weak,yes
5  Rain,Mild,High,Weak,yes
6  Rain,Cool,Normal,Weak,yes
7  Rain,Cool,Normal,Strong,no
8  Overcast,Cool,Normal,Strong,yes
9  Sunny,Mild,High,Weak,no
10 Sunny,Cool,Normal,Weak,yes
11 Rain,Mild,Normal,Weak,yes
12 Sunny,Mild,Normal,Strong,yes
13 Overcast,Mild,High,Strong,yes
14 Overcast,Hot,Normal,Weak,yes
15 Rain,Mild,High,Strong,no
```

```python
import pandas as pd
data = pd.read_csv("Play.csv")
print(data)
```

	Outlook	Temp	Humi	Wind	play
0	Sunny	Hot	High	Weak	no
1	Sunny	Hot	High	Strong	no
2	Overcast	Hot	High	Weak	yes
3	Rain	Mild	High	Weak	yes
4	Rain	Cool	Normal	Weak	yes
5	Rain	Cool	Normal	Strong	no
6	Overcast	Cool	Normal	Strong	yes
7	Sunny	Mild	High	Weak	no
8	Sunny	Cool	Normal	Weak	yes
9	Rain	Mild	Normal	Weak	yes
10	Sunny	Mild	Normal	Strong	yes
11	Overcast	Mild	High	Strong	yes
12	Overcast	Hot	Normal	Weak	yes
13	Rain	Mild	High	Strong	no

先验概率和条件概率的公式回顾

先验概率和条件概率的公式

$$P(Y=c_k)=\frac{\sum_{i=1}^{N} I(y_i=c_k)}{N}$$

$$P\left(X^{(j)}=a_{jl}\mid Y=c_k\right)=\frac{\sum_{i=1}^{N} I\left(x_i^{(j)}=a_{jl}, y_i=c_k\right)}{\sum_{i=1}^{N} I(y_i=c_k)}$$

- 指标 i 为样本指标, $x_i^{(j)}$ 表示第 i 个样本的第 j 个特征
- 其中 a_{jl} 表示第 j 个特征的第 l 个选择

先验概率和条件概率的计算

先验概率 $P(play = tag)$ 的计算

链接

- 计算 tag = yes 的概率
- 计算 tag = no 的概率

条件概率 $P(attr = value|play = tag)$ 的计算步骤

链接

1. 获取 attr 对应的列 A
2. 获取 play 对应的列 C
3. 通过 A == value 获取 A 对应位置等 (True) 或不等 (False) 于 value 的结果 A0, 通过 C == tag 获取 C 对应位置等 (True) 或不等 (False) 于 tag 的结果 C0.
4. 计算 A0 和 C0 对应位置同时为 True 的次数 (sum(A0 & C0)), n_0, 计算 C0 为 True 的次数 n, 返回条件概率值 $\frac{n_0}{n}$

主要内容

主函数实现-明确输入输出和处理步骤

主函数实现

首先回答以下问题

1. 主函数要实现的功能
   - 给定数据集, 各个属性名称及对应的值 (字符串 (str)), 以及要计算后验概率的标签 (字符串 (str)), 计算对应的后验概率.
2. 主函数的输入和输出是什么?
   - 输入: 数据集, 各个属性名称 (字符串型 (str)) 及对应的值 (字符串 (str)), 计算后验概率的标签 (字符串 (str))
   - 输出: 后验概率 (浮点型)
3. 主函数的主要处理步骤有哪些?

Algorithm 1 计算 $P(play = yes|Outlook = Sunny, Temp = Cool, Humi = High, Wind = Strong)$

Input: 数据集, 各个属性名称 (字符串型 (str)) 及对应的值 (字符串 (str)), 以及要计算后验概率的标签 (字符串 (str)).

1: 获取标签列的标签集合 tagset
2: 初始化空字典用于存储各标签对应的联合概率.
3: **for** tag in tagset **do**
4: 计算联合概率 P(Outlook = Sunny, Temp = Cool, Humi = High, Wind = Strong, play = tag)
5: 更新标签 tag 对应的概率值
6: **end for**
7: **Output:** 给定标签的后验概率值 (浮点型).

主函数的实现-以终为始

用户如何使用这个函数呢?

- 将属性名称和对应的属性值分别放在一个元组里

```
p1 = Posterior(data, "yes", attrs=("Outlook", "Temperature", "Humidity", "Wind"),
                                        values=("Sunny","Cool","High","Strong"))
```

- 按参数关键字调用函数

```
p1 = Posterior(data, "yes", Outlook="Sunny", Temperature="Cool", Humidity="High",
                                        Wind="Strong")
```

对应的函数定义方式

- 将属性名称和对应的属性值分别放在一个元组里

```
def Posterior(data, label, attrs, values): pass
```

- 按参数关键字调用函数

```
def Posterior(data, label, **argv): pass
```

主函数的实现代码

采用字典参数的方式实现

```
def Posterior(data, label, **argv):
    Y = data.iloc[:,-1]#约定最后一列为标签列
    tagset = set(Y) #获取最后一列标签列的标签集合
    probs = {} #初始化空字典用于存储各标签对应的联合概率
    for tag in tagset:
        p = ProbabilityJoin(data, tag, **argv)
        probs[tag] = p
    #1. 获取字典值的集合 2. 对所获取的集合求和
    pjoin = sum(probs.values())
    return {key : probs[key] / pjoin for key in probs}[label]
```

- ProbabilityJoin是尚未实现的函数, 先假设我们以这样的方式来使用这个函数, 它需要的输入也是数据集, 标签以及属性名称和对应值
- 重复我们写 Posterior 函数的过程实现ProbabilityJoin

函数 ProbabilityJoin 实现-明确输入输出和处理步骤

子函数 ProbabilityJoin 实现

首先回答以下问题

1. 子函数要实现的功能
   - 给定数据集, 各个属性名称及对应的值 (字符串 (str)), 以及要计算后验概率的标签 (字符串 (str)), 计算对应的联合概率.
2. 子函数的输入和输出是什么?
   - 输入: 数据集, 各个属性名称 (字符串型 (str)) 及对应的值 (字符串 (str)), 计算后验概率的标签 (字符串 (str))
   - 输出: 联合概率 (浮点型)
3. 子函数的主要处理步骤有哪些?

Algorithm 2 计算联合概率 $P(Outlook = Sunny, Temp = Cool, Humi = High, Wind = Strong, tag = yes)$

Input: 数据集, 各个属性名称 (字符串型 (str)) 及对应的值 (字符串 (str)), 以及要计算后验的标签 (字符串 (str)).

```
获取标签列 Y
计算 tag 对应的先验概率, 作为 tag 对应联合概率的初始值 P
for attr in argv.keys() do
    计算条件概率 P(attr = argv[attr]|Y = tag) p
    更新标签 tag 对应的概率值
end for
Output: 给定标签的联合概率值 (浮点型).
```

函数 ProbabilityJoin 实现-以终为始

用户如何使用这个函数?

- 在主函数 Posterior 里面调用

```
p = ProbabilityJoin(data, tag, **argv)
```

- 手动调用

```
p = ProbabilityJoin(data, "yes", Outlook = "Sunny", Temp = "Cool", Humi
                                        = "High", Wind = "Strong")
```

对应的函数定义

```
def ProbabilityJoin(data, label, **argv):
    pass
```

函数 ProbabilityJoin 实现代码

采用字典参数的方式实现 *s4f51.py*

```
def ProbabilityJoin(data, label, **argv):
    Y = data.iloc[:,-1] #约定最后一列作为标签列
    P = sum(Y == label)/len(Y)
    for attr in argv.keys():
        p = ProbabilityOnY(data, label, attr, argv[attr])
        P *= p
    return P
```

- ProbabilityOnY是尚未实现的函数, 假设以此方式使用函数, 它需要的输入是数据集, 标签, 要计算条件概率的属性名称和属性值.
- 重复函数 ProbabilityJoin 实现过程实现ProbabilityOnY

函数 ProbabilityOnY 实现

实现思路及代码 *s4f61.py*

- Pr(Outlook = “Sunny” | y = “yes”) = Pr(Outlook = “Sunny”, y = “yes”) / Pr(y = “yes”)

```
def ProbabilityOnY(data, label, attr, value):
    Y = data.iloc[:,-1] == label
    A = data[attr] == value
    return sum(Y & A) / sum(Y)
```

完整代码参考

s4f71.py

```
def Posterior(data, label, **argv):
    Y = data.iloc[:,-1]#约定最后一列为标签列
    tagset = set(Y) #获取最后一列标签列的标签集合
    probs = {} #初始化空字典用于存储各标签对应的联合概率
    for tag in tagset:
        p = ProbabilityJoin(data, tag, **argv)
        probs[tag] = p
    pjoin = sum(probs.values()) #1. 获取字典值的集合 2. 对所获取的集合求和
    return {key : probs[key] / pjoin for key in probs}[label]
def ProbabilityJoin(data, label, **argv):
    Y = data.iloc[:,-1] #约定最后一列作为标签列
    P = sum(Y == label)/len(Y)
    for attr in argv.keys():
        p = ProbabilityOnY(data, label, attr, argv[attr])
        P *= p
    return P
def ProbabilityOnY(data, label, attr, value):
    Y = data.iloc[:,-1] == label
    A = data[attr] == value
    return sum(Y & A) / sum(Y)
if __name__ == '__main__':
    data = pd.read_csv("Play.txt")
    p1=Posterior(data,"yes",Outlook="Sunny",Temp="Cool",Humi="Normal",Wind="Weak")
```

拉普拉斯平滑

拉普拉斯平滑

$$P_\lambda\left(Y=c_k\right)=\frac{\sum_{i=1}^{N} I\left(y_i=c_k\right)+\lambda}{N+K\lambda}$$

$$P_\lambda\left(X^{(j)}=a_{jl} \mid Y=c_k\right)=\frac{\sum_{i=1}^{N} I\left(x_i^{(j)}=a_{jl}, y_i=c_k\right)+\lambda}{\sum_{i=1}^{N} I\left(y_i=c_k\right)+S_j\lambda}$$

- 指标 i 为样本指标, $x_i^{(j)}$ 表示第 i 个样本的第 j 个特征
- 其中 a_{jl} 表示第 j 个特征的第 l 个选择, S_j 表示第 j 个特征的个数, K 代表种类的个数
- $\lambda=1$ 对应拉普拉斯平滑

引入拉普拉斯平滑

条件概率函数 *s4f91.py*

```
def ProbabilityOnY(data, label, attr, value, lambda = 1):
    Y = data.iloc[:,-1] == label
    A = data[attr] == value
    S = len(set(data[attr]))
    return (sum(Y & A) + lambda) / (sum(Y) + lambda * S)
```

联合概率函数计算 *s4f92.py*

```
def ProbabilityJoin(data, label, **argv, lambda = 1):
    Y = data.iloc[:,-1] #约定最后一列作为标签列
    K = len(set(Y))
    P = (sum(Y == label) + lambda)/ (len(Y) + K * lambda)
    for attr in argv.keys():
        p = ProbabilityOnY(data, label, attr, argv[attr])
        P *= p
    return P
```

多项式朴素贝叶斯优化

优化动机

- 根据不同的天气组合估算后延概率值时, 要计算 P(attr=value|Y=label)
- label 和 value 的组合可能多次出现, 例如第 1,2 个样本都会计算 P(Outlook = Sunny|play = no), P(Temperature = Hot|play = no) 以及 P(Humidity = High|play = no)
- 优化函数的实现, 避免重复计算

优化思路

1. 将计算过的条件概率值保存下来
2. 确定 key, 如果 key 没有出现过, 则保存 key 与对应的条件概率值
3. key 应该如何选择?

链接

条件概率实现优化

利用默认参数字典

```
def ProbabilityOnY(data, label, attr, value, lambda = 1, memo = {}):
    Y = data.iloc[:,-1] == label
    A = data[attr] == value
    S = len(set(data[attr]))
    return (sum(Y & A) + lambda) / (sum(Y) + lambda * S)
def ProbabilityJoin(data, label, **argv, lambda = 1, memo = {}):
    Y = data.iloc[:,-1] #约定最后一列作为标签列
    K = len(set(Y))
    P = (sum(Y == label) + lambda)/ (len(Y) + K * lambda)
    for attr in argv.keys():
        p = ProbabilityOnY(data, label, attr, argv[attr])
        P *= p
    return P
```

链接

主要内容

主要内容

高斯朴素贝叶斯

X 为连续变量

性别	身高 (英尺)	体重 (磅)	脚掌 (英寸)
男	6	180	12
男	5.92	190	11
男	5.58	170	12
男	5.92	165	10
女	5	100	6
女	5.5	150	8
女	5.42	130	7
女	5.75	150	9

1. 已知某人身高 6 英尺、体重 130 磅，脚掌 8 英寸: 推断某人性别
2. 给定性别, 条件概率如何计算?
   - Pr(身高 | 性别), Pr(体重 | 性别), Pr(脚掌 | 性别)

高斯朴素贝叶斯

给定类别 $Y = c$ 时, 假设连续变量 x 服从高斯分布

$$p(x|c) \sim \mathcal{N}(\mu_c, \sigma_c^2)$$

- 参数 μ_c, σ_c^2 分别为类别 c 对应样本 x 的均值和方差.

高斯朴素贝叶斯实现的关键步骤 *s0f11.py*

1. 提取对应给定类标签连续属性值.
2. 计算均值 μ 和方差 σ^2
3. 计算 x 在 $N(\mu, \sigma^2)$ 的概率密度值.

```
def ProbabilityOnY(data, label, attr, value):
    Y = data.iloc[:,-1] == label
    X = data[attr][Y]
    mu = np.mean(X)
    sigma = np.std(X)
    return prob(mu, sigma)(value) #prob根据mu, sigma生成概率密度函数
```

主要内容

高斯朴素贝叶斯实现的优化

如何避免反复进行概率密度函数的估计 *s1f01.py*

```
def ProbabilityOnY(data, label, attr, value, memo = {}):
    Y = data.iloc[:,-1] == label
    X = data[attr][Y]
    mu = np.mean(X)
    sigma = np.std(X)
    return prob(mu, sigma)(value) #prob根据mu, sigma生成概率密度函数
```

- 选取什么作为 key? 对应的 value 是什么?

主要内容

主要内容

离散和连续性变量共存

多项式朴素贝叶斯和高斯朴素贝叶斯统一实现 *s0f01.py*

```
def Posterior(data, label, **argv):
    Y = data.iloc[:,-1]#约定最后一列为标签列
    tagset = set(Y) #获取最后一列标签列的标签集合
    probs = {} #初始化空字典用于存储各标签对应的联合概率
    for tag in tagset:
        p = ProbabilityJoin(data, tag, **argv)
        probs[tag] = p
    pjoin = sum(probs.values()) #1. 获取字典值的集合 2. 对所获取的集合求和
    return {key : probs[key] / pjoin for key in probs}[label]
def ProbabilityJoin(data, label, **argv):
    Y = data.iloc[:,-1] #约定最后一列作为标签列
    P = sum(Y == label)/len(Y)
    for attr in argv.keys():
        p = ProbabilityOnY(data, label, attr, argv[attr])
        P *= p
    return P
def ProbabilityOnY(data, label, attr, value):
    Y = data.iloc[:,-1] == label
    A = data[attr] == value
    return sum(Y & A) / sum(Y)
if __name__ == '__main__':
    data = pd.read_csv("Play.txt")
    p1=Posterior(data,"yes",Outlook="Sunny",Temp="Cool",Humi="Normal",Wind="Weak")
```

朴素贝叶斯类

朴素贝叶斯类 *s0f12.py*

```
class NaiveBayes:
    def __init__():
        pass
    def fit(self, X, Y):
        pass
    def predict(self, x, **argv):
        pass
    def score(self, tX, tY):
        pass
```

- fit 函数: X 为训练样本, Y 为对应的标签列, 拟合朴素贝叶斯模型, 用于后面的 predict 和 score
- predict 函数: x 可以为 pandas.Series 或 pandas.DataFrame, 关键字参数允许类似 model.predict(outlook = “Sunny”, Temp = “Hot”, Humi = “High”, Wind = “Weak”) 的调用
- score 函数: 根据测试样本的 tX 预测对应的 Y, 并于 tY 进行比较计算分数, criteria 取值分别为 accuracy, auc, 对应计算准确率或 auc.

本周题目汇总

1. **week13nb–s2f0–r152**
   在下述一些量的计算中, 变化的是什么, 不变的是什么? 还可以对过程进行怎样的归纳?
2. **week13nb–s3f4–r165**
   实现函数, 输入标签列及对应的 tag, 输出对应的概率
3. **week13nb–s3f4–r924**
   实现函数, 输入标签列及对应的 tag, 输入属性列和对应的 value, 输出对应的条件概率
4. **week13nb–s4f10–r37**
   在计算条件概率时, key 应该如何选择?
5. **week13nb–s4f11–r546**
   完成下面的代码