第7讲 Naïve Bayes

- 概率基础
- 贝叶斯定理

2019-04-10

- Naïve Bayes分类器
- 实践：Python

概率基础

- 联合概率(joint probability)

  表示*A*事件和*B*事件同时发生的概率， $P(A \cap B)$

- 边际概率(marginal probability)

  在*A*和*B*的样本空间中，只看*A*或*B*的概率，称之边际概率

- 条件概率(conditional probability)

  在发生*A*的条件下，发生*B*的概率，称为$P(B|A)$

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

概率基础

	赞成(*B*1)	反对(*B*2)	合计
男性(*A*1)	40	**120**	**160**
女性(*A*2)	10	**30**	**40**
合计	50	**150**	**200**

2019-04-10

联合概率：P(男性，赞成) = P(A1∩B1) = 40/200 = 0.2

边际概率： P(赞成) = P(B1) = P(A1∩B1) + P(A2∩B1) = 0.25

条件概率： P(赞成|男性) = P(B1|A1) = P(A1∩B1) / P(A1) = 0.25

概率基础

- 乘法法则(Multiplicative rule)

$$P(A\cap B)=P(B)\times P(A|B)=P(A)\times P(B|A)$$

- 独立事件 2019-04-10

设事件 A 和事件 B 满足以下条件：

$$P(A\cap B)=P(A)\times P(B)$$

或： $P(A)>0,\ P(B|A)=P(B)$

$P(B)>0,\ P(A|B)=P(A)$

则称 A 与 B 为『独立事件』

Naïve Bayes分类器

- 一个数据样本包含n个特征，即$X = \{x_1, x_2, \ldots, x_n\}$，可属于$m$个不同类别之一，即$C = \{c_1, c_2, \ldots, c_m\}$
- X 属于类别c_i的概率：

$$P(C_i \mid X) = \frac{P(X \mid C_i)P(C_i)}{P(X)}$$

2019-04-10

- 条件独立假设（在给定类别情况下，所有特征相互独立）

$$P(X \mid C_i) = \prod_{d=1}^{n} P(x_d \mid c_i)$$

- X的类别：$X \in C_k$，$P(C_k|X) = \max\{P(C_1|X), P(C_2|X), \ldots, P(C_m|X)\}$

例1: 判断是否购买计算机

age	income	student	credit_rating	buys_computer
youth	high	no	fair	no
youth	high	no	excellent	no
middle_aged	high	no	fair	yes
senior	medium	no	fair	yes
senior	low	yes	fair	yes
senior	low	yes	excellent	no
middle_aged	low	yes	excellent	yes
youth	medium	no	fair	no
youth	low	yes	fair	yes
senior	medium	yes	fair	yes
youth	medium	yes	excellent	yes
middle_aged	medium	no	excellent	yes
middle_aged	high	yes	fair	yes
senior	medium	no	excellent	no

2019-04-10

例1: 判断是否购买计算机

- **X = (age=youth, income=medium, student=yes, credit_rating=fair)**
  buy computer?
- **P(buy=yes|X) > P(buy=no|X) ?**

2019-04-10

- P(buy=yes) = 9/14 = 0.643, P(buy=no) = 5/14 = 0.357
- P(age=youth|buy=yes) = 2/9 = 0.222
- P(X|buy=yes) = P(age=youth|buy=yes) * P(income=medium|buy=yes) * P(student=yes|buy=yes) * P(credit=fair|but=yes) = 0.044
- P(X|buy=no) = 0.019
- P(buy=yes|X) = P(X|buy=yes) * P(buy=yes) = 0.028
- P(buy=no|X) = P(X|buy=no) * P(buy=no) = 0.007

零概率问题和平滑方法

- 零概率问题：在计算实例的概率时，如果某个量x，在训练集中没有出现过，会导致整个实例的概率结果为0。
- 拉普拉斯平滑（Laplace Smoothing）：最早提出用加1的方法估计没有出现过的事件的概率

2019-04-10

- 应用举例

  假设在文本分类中，有3个类：C1、C2、C3，在指定的1000个训练样本中，某个词语k1在各个类别中观测计数分别为0，990，10，则k1的概率为0，0.99，0.01

  拉普拉斯平滑：1/1003 = 0.001，991/1003=0.988，11/1003=0.011

- 实际使用中也经常使用加λ（1≥λ≥0）来代替简单加1，如果对*N*个计数都加上λ，这时分母加上*k**λ，*k*为类别数

例2：判断性别？

- 通过某人的特征数据，包括身高、体重、脚的尺寸，判定其性别
- 训练数据集：

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

2019-04-10

- 特征是连续值，假设训练集样本的各个特征值服从高斯分布

例2：判断性别？

- 特征为连续值，假定特征服从高斯分布
- 高斯分布的概率密度函数

$$g(x,\ \mu,\ \sigma)=\frac{1}{\sqrt{2\pi}\sigma}e^{\frac{-(x-\mu)^2}{2\sigma^2}}$$

用来反映各特征值的相对可能性

$$P(X\,|\,c_i)=g(x_k,\mu_{c_i},\sigma_{c_i})$$

- 计算训练样本集中不同类别中，不同特征值：均值，方差

例2：判断性别？

性别	均值(身高)	方差(身高)	均值(体重)	方差(体重)	均值(脚的尺寸)	方差(脚的尺寸)
男性	5.855	3.5033e-02	176.25	1.2292e+02	11.25	9.1667e-01
女性	5.4175	9.7225e-02	132.5	5.5833e+02	7.5	1.6667e+00

预测问题：某人身高6英尺，体重130磅，脚的尺寸为8英尺，请推断某人是男性还是女性？

- 假定 $P(male) = P(female) = 0.5$
- $posterior(male) = P(male)p(height|male)p(weight|male)p(footsize|male)/\mathbf{evidence}$
- $posterior(female) = P(female)p(height|female)p(weight|female)p(footsize|female)/\mathbf{evidence}$

$$P(height \mid male) = \frac{1}{\sqrt{2\pi\alpha}} e^{\frac{-(x-\mu)^2}{2\alpha^2}}$$

$\mathbf{posterior(male) = 6.1984e^{-09}}$

$\mathbf{posterior(female) = 5.3778e^{-04}}$

Naïve Bayes分类器类型

- 特征是离散变量，可以用频率来估计概率
- 特征是连续变量，通常假定连续变量服从高斯分布
- Scikit-learn的Naïve Bayes模型：
  - BernoulliNB — 伯努利模型
  - GaussianNB — 高斯模型
  - MultinomialNB — 多项式模型

2019-04-10

伯努利Naïve Bayes

- **BernoulliNB：适合伯努利分布的数据集**
- **伯努利分布：也称“二项式分布”或者“0-1分布”，即做*n*次伯努利试验，规定每次试验的结果只有两个**

2019-04-10

- **离散值特征，且取值只能是1和0**

伯努利Naïve Bayes

```
import numpy as np
# 将X，y赋值为np数组
# X包含4个特征：刮风，闷热，多云，天气预报有雨。
X = np.array([[0, 1, 0, 1],
              [1, 1, 1, 0],
              [0, 1, 1, 0],
              [0, 0, 0, 1],
              [0, 1, 1, 0],
              [0, 1, 0, 1],
              [1, 0, 0, 1]])
# y为样本标签，0代表没下雨，1代表下雨
y = np.array( [0, 1, 1, 0, 1, 0, 0])
```

```
# 构建BernoulliNB分类器
from sklearn.naive_bayes import BernoulliNB
clf = BernoulliNB()
clf.fit(X, y)

# 进行预测
Next_Day = [0, 1, 1, 0]
pre = clf.predict(Next_Day)
if pre == [1]:
    print("下雨")
else:
    print("无雨")
```

下雨

2019-04-10

```
clf.predict_proba(Next_Day)
```

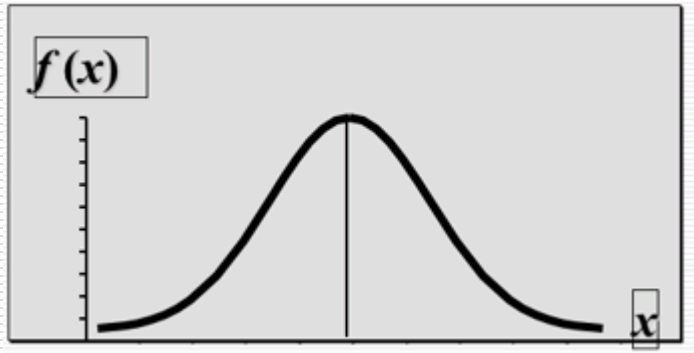
```
array([[ 0.13848881,  0.86151119]])
```

高斯Naïve Bayes

- GaussianNB：适合高斯分布的数据集
- 高斯分布：也称正态分布(normal distribution)
- 连续值特征，概率密度函数：

2019-04-10

$$f(x)=\frac{1}{\sigma\sqrt{2\pi}}e^{\frac{1}{2\sigma^2}(x-\mu)^2},\quad -\infty<x<+\infty$$



高斯Naïve Bayes

```
# 导入数据集生成工具
from sklearn.datasets import make_blobs
# 生成500个样本数据，5个类别
X, y = make_blobs(n_samples=500, centers=5,random_state=8)
```

```
print(X)
```

```
[[ -4.43344765e+00  -9.14511574e+00]
 [ -5.06998128e+00  -9.75464122e+00]
 [  6.54464509e+00   8.99873511e-01]
 [  3.25023324e-01   1.50633915e-01]
 [ -1.51028157e+00  -1.10581275e+00]
 [ -8.90489310e+00  -1.10427432e+01]
 [  9.28383472e-02  -2.00771121e-02]
 [ -6.21720086e+00  -1.11227678e+01]
 [  7.63027116e+00   8.69797933e+00]
 [  7.92430026e+00   1.04511206e-01]
```

```
print(y)
```

```
[2 2 4 3 4 3 4 1 3 0 2 0 0 2 3 0 2 3 0 4 1 3 4 1 4 4 2 3 3 1 0 0 4 1 2
 0 2 4 4 3 1 0 3 0 0 3 3 2 2 4 4 4 2 0 3 1 0 0 4 0 4 3 2 0 0 3 2 0 0 0 3 2
 1 1 3 2 0 3 1 3 1 3 3 2 3 3 1 4 0 2 4 3 3 4 1 2 4 2 3 2 2 1 3 1 2 0 0 4 1
 4 4 0 2 1 0 4 0 1 0 0 0 3 0 2 0 3 1 3 0 1 2 0 3 4 1 1 2 1 2 1 2 2 3 2 0 4
 1 0 2 0 1 1 4 1 2 2 2 4 3 3 0 2 2 4 4 4 0 1 4 1 0 0 2 0 3 4 3 0 0 1 3 4 4
 0 2 4 4 3 0 1 1 1 4 2 0 1 4 2 2 1 3 1 3 4 3 3 2 3 3 2 3 4 3 3 1 1 3 0 4 4
 4 1 3 0 4 4 0 2 4 1 0 2 4 0 1 3 4 0 4 1 3 3 1 1 3 3 1 4 3 2 3 0 2 2 1 1 1
 2 2 0 1 2 4 3 0 2 3 0 2 0 4 4 1 0 0 4 1 1 1 4 3 0 1 2 1 1 0 3 3 2 2 1 4 3
 4 4 2 0 1 4 3 0 3 0 2 3 2 1 4 0 1 3 2 2 3 2 1 2 1 1 4 4 1 3 4 2 4 4 0 1 3
 0 1 1 2 1 3 4 0 2 2 3 1 4 2 3 1 1 4 0 3 4 0 2 3 2 0 1 1 3 3 0 2 2 1 2 0 0
 2 4 0 1 2 2 1 0 4 3 1 1 2 2 1 1 2 1 0 0 3 2 2 4 4 2 2 3 1 0 4 0 4 3 3 0 3
 1 0 1 4 4 4 1 3 0 2 2 4 0 3 2 2 1 4 4 4 0 4 3 2 4 3 3 1 3 0 0 2 0 0 3 3 2
 2 3 1 4 0 4 3 2 4 0 3 3 3 0 0 4 3 1 1 4 3 3 3 2 2 1 4 2 4 0 1 0 4 4 0 4 4
 3 3 4 2 4 1 4 0 1 0 1 4 4 4 1 2 1 1 3]
```

高斯Naïve Bayes

```
# 导入数据集生成工具
from sklearn.datasets import make_blobs
X, y = make_blobs(n_samples=500, centers=5,random_state=8)
# 导入数据集拆分工具
from sklearn.model_selection import train_test_split
# 生成500个样本数据，5个类别
# 数据拆分成训练集和测试集
X_train,X_test,y_train,y_test=train_test_split(X,y,random_state=8)
# 构建 GaussianNB 分类器
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
gnb.fit(X_train, y_train)
print('模型得分：{:.3f}'.format(gnb.score(X_test, y_test)))
```

```
模型得分：0.968
```

2019-04-10

多项式Naïve Bayes

- MultinomialNB：适合多项式分布的数据集
- 多项式分布(Multinomial Distribution)：二项式分布的推广。做n次试验，每次结果有m（>2）个，且 m 个结果发生的概率互斥且和为1，则发生其中一个结果 X 的概率
- 离散型特征值

$$P(X_1=x_1,X_2=x_2,\cdots,X_k=x_k)=\frac{n!}{x_1!x_2!\cdots x_k!}p_1^{x_1}p_2^{x_2}\cdots p_k^{x_k}$$

```
import numpy as np
X = np.random.randint(5, size=(6, 100))
y = np.array([1, 2, 3, 4, 5, 6])
from sklearn.naive_bayes import MultinomialNB
clf = MultinomialNB()
clf.fit(X, y)
MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True)
print(clf.predict(X[2:3]))
```

```
[3]
```

Naïve Bayes应用过程

（1）收集数据：使用任何方法
（2）准备数据：数值型或者布尔型数据
（3）分析数据：使用任意方法
（4）训练数据：计算不同的独立特征的条件概率
（5）测试数据：计算错误率
（6）使用算法：常用文档分类， 也可用在任意分类场景

2019-04-10

Naïve Bayes 进行文本分类

1. 准备数据：从文本中构建词向量
2. 训练算法：通过朴素贝叶斯算法统计出所有词向量的各种分类的概率

2019-04-10

3. 测试过程：对于待分类的文档，在转换为词向量之后，从训练集中取得该词向量的各种分类概率，概率最大的类别作为分类结果

Naïve Bayes 进行文本分类

```python
from numpy import *

# 创建实验样本集函数
def loadDataSet():
    postingList=[['my', 'dog', 'has', 'flea', 'problems', 'help', 'please'],
                 ['maybe', 'not', 'take', 'him', 'to', 'dog', 'park', 'stupid'],
                 ['my', 'dalmation', 'is', 'so', 'cute', 'I', 'love', 'him'],
                 ['stop', 'posting', 'stupid', 'worthless', 'garbage'],
                 ['mr', 'licks', 'ate', 'my', 'steak', 'how', 'to', 'stop', 'him'],
                 ['quit', 'buying', 'worthless', 'dog', 'food', 'stupid']]

    # 标签：1代表侮辱，0代表正常
    classVec = [0, 1, 0, 1, 0, 1]
    return postingList,classVec

# 创建词汇表函数
def createVocabList(dataSet):
    # 利用集合内元素的唯一性创建一个包含所有词汇的词表

    # 创建一个空集
    vocabSet = set([])
    for document in dataSet:
        # 两个集合的并集
        vocabSet = vocabSet | set(document)
    return list(vocabSet)
```

2019-04-10

```python
# 创建实验样本集
listOPost, listClasses = loadDataSet()

# 创建词汇表(无重复元素)
myVocabList = createVocabList(listOPost)

print('listOPost: ',listOPost)
print('listClasses: ',listClasses)

print('myVocabList: ',myVocabList)

print('len(listOPost)= ',len(listOPost))
print('len(myVocabList)= ',len(myVocabList))
```

```
listOPost:  [['my', 'dog', 'has', 'flea', 'problems', 'help', 'p
y', 'dalmation', 'is', 'so', 'cute', 'I', 'love', 'him'], ['stop
y', 'steak', 'how', 'to', 'stop', 'him'], ['quit', 'buying', 'wo
listClasses:  [0, 1, 0, 1, 0, 1]
myVocabList:  ['my', 'ate', 'flea', 'how', 'buying', 'has', 'ste
'I', 'not', 'is', 'posting', 'worthless', 'so', 'stupid', 'dog',
len(listOPost)=  6
len(myVocabList)=  32
```

Naïve Bayes 进行文本分类

```
# 文档词集模型：向量中的元素值是1或0，分别表示词汇表中的单词在输入文档中是否出现
def setOfWords2Vec(vocabList, inputSet):
    # vocablist为词汇表，inputSet为输入邮件

    # 创建一个与词汇表等长的向量，并将其元素都设为0
    returnVec = [0] * len(vocabList)
    for word in inputSet:
        if word in vocabList:
            # 查找单词的索引
            returnVec[vocabList.index(word)] = 1
        else:
            print("the word: %s is not in my vocabulary" %word)
    return returnVec
```

2019-04-10

```
# 调用setOfWords2Vec()函数生成词集向量：

# 构建listOPost列表0位置对应的词向量
setOfWords2Vec0 = setOfWords2Vec(myVocabList, listOPost[0])
print(setOfWords2Vec0)

# 构建listOPost列表3位置对应的词向量
setOfWords2Vec3 = setOfWords2Vec(myVocabList, listOPost[3])
print(setOfWords2Vec3)
```

```
[1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1]
```

Naïve Bayes 进行文本分类

```
# 构建朴素贝叶斯分类器训练函数

def train(trainMat,trainCategory):
    #trainMat为训练样本集的词向量矩阵，每一行为一个邮件的词向量
    #trainCategory为对应的类别标签（0表示正常，1表示垃圾）

    numTrain = len(trainMat)
    #is vocabulary length
    numWords = len(trainMat[0])
    pAbusive = sum(trainCategory) / float(numTrain)
    p0Num = zeros(numWords); p1Num = zeros(numWords)
    #初始化概率
    p0Denom = 0.0; p1Denom = 0.0
    for i in range(numTrain):
        if trainCategory[i] == 1:
            # 统计类1中每个单词的个数
            p1Num += trainMat[i]
            # 类1的单词总数
            p1Denom += sum(trainMat[i])
        else:
            # 统计类0中每个单词的个数
            p0Num += trainMat[i]
            # 类0的单词总数
            p0Denom += sum(trainMat[i])
    # 类1中每个单词的概率
    p1Vec = (p1Num / p1Denom)
    # 类0中每个单词的概率
    p0Vec = (p0Num / p0Denom)
    return p0Vec,p1Vec,pAbusive
```

```
# 调用train()函数，返回两个概率向量和一个概率值

listOPost,listClasses = loadDataSet()
myVocabList = createVocabList(listOPost)

# for循环使用词向量充填trainMat列表
trainMat = []
for postinDoc in listOPost:
    trainMat.append(setOfWords2Vec(myVocabList, postinDoc))
p0V,p1V,pAb = trainNB0(trainMat, listClasses)
```

2019-04-10

```
print('p0V =' , p0V)
print('p1V =' , p1V)
print('pAb =' , pAb)
```

```
p0V = [ 0.125       0.04166667  0.04166667  0.04166667  0.          0.04166667
  0.04166667  0.08333333  0.          0.04166667  0.          0.
  0.04166667  0.04166667  0.04166667  0.04166667  0.          0.04166667
  0.          0.          0.04166667  0.          0.04166667  0.04166667
  0.          0.04166667  0.04166667  0.          0.04166667  0.
  0.04166667  0.04166667]
p1V = [ 0.          0.          0.          0.          0.05263158  0.          0.
  0.05263158  0.05263158  0.          0.05263158  0.05263158  0.          0.
  0.05263158  0.          0.05263158  0.          0.05263158  0.10526316
  0.          0.15789474  0.10526316  0.          0.05263158  0.          0.
  0.05263158  0.          0.05263158  0.          0.05263158]
pAb = 0.5
```

Naïve Bayes 进行文本分类

```
# 构造分类函数

def classfy(vec2classfy, p0Vec, p1Vec, pClass1):
    p1 = sum(vec2classfy * p1Vec) + log(pClass1)
    p0 = sum(vec2classfy * p0Vec) + log(1-pClass1)
    if p1 > p0:
        return 1
    else:
        return 0

def testingNB():
    listOPosts, listClasses = loadDataSet()
    myVocabList = createVocabList(listOPosts)
    trainMat = []
    for postinDoc in listOPosts:
        trainMat.append(setOfWords2Vec(myVocabList, postinDoc))
    p0V, p1V, pAb = train(array(trainMat), array(listClasses))
    testEntry = ['love', 'my', 'dalmation']
    thisDoc = array(setOfWords2Vec(myVocabList, testEntry))
    print (testEntry, 'classified as: ', classfy(thisDoc, p0V, p1V, pAb))
    testEntry = ['stupid', 'garbage']
    thisDoc = array(setOfWords2Vec(myVocabList, testEntry))
    print (testEntry, 'classified as: ', classfy(thisDoc, p0V, p1V, pAb))
```

2019-04-10

```
testingNB()
```

```
['love', 'my', 'dalmation'] classified as:  0
['stupid', 'garbage'] classified as:  1
```

Naïve Bayes优缺点

- 计算速度快
- 规则清楚易懂
- 仅能应用于分类问题 2019-04-10
- 假设变量间相互独立，但实际中变量存在相关性

Bayes Classifier

- 朴素贝叶斯分类假定类条件独立，即给定样本的类标号，属性的值相互条件独立，也因此称为“朴素”。
- 实践中，变量之间的依赖可能存在。贝叶斯信念网络描述联合条件概率分布，它允许在变量的子集间定义类条件独立性。它提供一种因果关系的图形。
- 贝叶斯信念网络可用于分类，它是图形模型。相比朴素贝叶斯，它能够处理属性子集间有依赖关系的分类。

2019-04-10