Title

April 30, 2019

1 贝叶斯分类器简介

贝叶斯分类器很简单，基本上就是一个贝叶斯公式，要理解透彻贝叶斯分类器需要搞清楚两个概念。

似然函数 在数理统计学中，似然函数是一种关于统计模型中的参数的函数，表示模型参数中的似然性。似然函数在统计推断中有重大作用，如在最大似然估计和费雪信息之中的应用等等。“似然性”与“或然性”或“概率”意思相近，都是指某种事件发生的可能性，但是在统计学中，“似然性”和“或然性”或“概率”又有明确的区分。概率用于在已知一些参数的情况下，预测接下来的观测所得到的结果，而似然性则是用于在已知某些观测得到的结果时，对有关事物的性质的参数进行估计。

在这种意义上，似然函数可以理解为条件概率的逆反。在已知某个参数B时，事件A会发生的概率写作：

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

利用贝叶斯定理，

$$P(B \mid A) = \frac{P(A \mid B)\ P(B)}{P(A)}$$

注意到这里并不要求似然函数满足归一性：$\sum_{b \in \mathcal{B}} P(A \mid B = b) = 1$ 。一个似然函数乘以一个正的常数之后仍然是是似然函数。对所有的$\alpha > 0$，都可以有似然函数：

$$L(b \mid A) = \alpha\ P(A \mid B = b)$$

最大似然估计 我们首先要定义似然函数:

$$\mathsf{lik}(\theta) = f_D(x_1, \ldots, x_n \mid \theta)$$

并且在θ的所有取值上通过令一阶导数等于零，使这个函数取到最大值。这个使可能性最大的$\widehat{\theta}$值即称为θ的最大似然估计。

1.1 例子

基本上就是似然函数就是一件事发生的概率公式，而最大似然估计是在当前似然函数下函数能取到的最大值，就是一件事最可能发生的概率，这在计算中很重要。
在一个垃圾邮件检测代码中贝叶斯公式的解释:

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

$P(B)$表示邮件是垃圾邮件的概率 垃圾邮件数/总邮件数
$P(A)$表示某个词在所有邮件中的出现概率：当前次出现次数/总次数
$P(A \mid B)$是垃圾邮件的情况下某个词出现的概率 是垃圾邮件某词出现次数/是垃圾邮件所有词数目
$P(B \mid A)$在一个词出现概率为A的情况下是垃圾邮件的概率 所有出现词的概率求和 对比垃圾邮件和非垃圾邮件那个值更大判断是否是垃圾邮件
看一个机器学习实战中贝叶斯检测垃圾词的代码

```
from numpy import *

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']]
    classVec = [0,1,0,1,0,1]    #1 is abusive, 0 not
    return postingList,classVec

def createVocabList(dataSet):
    vocabSet = set([])  #create empty set
    for document in dataSet:
        vocabSet = vocabSet | set(document) #union of the two sets
    return list(vocabSet)

def setOfWords2Vec(vocabList, inputSet):
    returnVec = [0]*len(vocabList)
    for word in inputSet:
        if word in vocabList:
            returnVec[vocabList.index(word)] = 1
        else: print "the word: %s is not in my Vocabulary!" % word
    return returnVecdef trainNB0(trainMatrix,trainCategory):
    numTrainDocs = len(trainMatrix)
    numWords = len(trainMatrix[0])
    pAbusive = sum(trainCategory)/float(numTrainDocs)
    p0Num = ones(numWords); p1Num = ones(numWords)      #change to ones()
```

```
    p0Denom = 2.0; p1Denom = 2.0                        #change to 2.0
    for i in range(numTrainDocs):
        if trainCategory[i] == 1:
            p1Num += trainMatrix[i]
            p1Denom += sum(trainMatrix[i])
        else:
            p0Num += trainMatrix[i]
            p0Denom += sum(trainMatrix[i])
    p1Vect = log(p1Num/p1Denom)          #change to log()
    p0Vect = log(p0Num/p0Denom)          #change to log()
    return p0Vect,p1Vect,pAbusive
def classifyNB(vec2Classify, p0Vec, p1Vec, pClass1):
    p1 = sum(vec2Classify * p1Vec) + log(pClass1)    #element-wise mult
    p0 = sum(vec2Classify * p0Vec) + log(1.0 - 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 = trainNB0(array(trainMat),array(listClasses))
    testEntry = ['love','my','dalmation']
    thisDoc = array(setOfWords2Vec(myVocabList, testEntry))
    print "p0v:", p0V, "p1V :", p1V, "pAb:", pAb
    print testEntry,'classified as: ',classifyNB(thisDoc,p0V,p1V,pAb)
    testEntry = ['stupid','garbage']
    thisDoc = array(setOfWords2Vec(myVocabList, testEntry))
    print testEntry,'classified as: ',classifyNB(thisDoc,p0V,p1V,pAb)
if __name__=='__main__':
    listOposts, listClasses = loadDataSet()
    myVocabList = createVocabList(listOposts)
    print setOfWords2Vec(myVocabList, listOposts[0])
    print testingNB()
```