贝叶斯定理及其应用

1 背景介绍

托马斯·贝叶斯(Thomas Bayes，约1701-1761)，英国数学家，出生于伦敦，做过神甫，1942年成为英国皇家学会会员，1761年4月7日逝世。贝叶斯在世时，并不为当时的人们所熟知，很少发表论文或出版著作，与当时学术界的人沟通交流也很少，所谓的贝叶斯定理其实是源于他生前为解决一个“逆向概率”问题写的一篇文章[[1]](#footnote-2)，而这篇文章是在他死后才由他的一位朋友发表出来的。

下面我们首先来看看什么是“逆向概率”问题。17世纪以前，人们对一件事情发生或不发生的概率，认为只有两种可能性，即要么发生，要么不发生，从来不会考虑某件事情发生的概率有多大，不发生的概率又有多大。后来进入18世纪以后，人们逐渐发现某件事情发生的概率应该是一个确切的数值，并逐步学会了“正向概率”计算，比如，“有一个袋子，里面装着N个白球和M个黑球，请问从袋子中取得白球的概率是多少？”。然而，如果问题变为“有一个袋子，里面装着若干个白球和黑球，请问从袋子中取得白球的概率是多少？”他们似乎就无能为力，只能告诉你概率是1/2，即要么取得白球，要么取不到白球。对此，贝叶斯有不同的看法，他认为“虽然我们不知道袋子里面白球和黑球的比例，但是如果我们可以摸出一个或多个球，观察这些取出来的球的颜色，那么我们是否可以就此计算我们抽取白球的概率呢？”这个问题就是所谓的“逆向概率”问题。

事实上，贝叶斯在解决“逆向概率”问题过程中，不再把抽取白球的概率当作一个确切的数值，而是把它看作一个随机变量，并指出我们应当利用新观察到的样本信息来修正以前对事物的认知，即我们应当从计算先验概率向计算后验概率转变。这种思维模式就称为贝叶斯思维模式，即：

* 先验分布样本信息后验分布

其中，先验分布是指在实验开始前定下的属于基本前提性质的分布，或称为的无条件分布。先验信息一般来源于经验跟历史资料。比如林丹跟某选手对决，解说一般会根据林丹历次比赛的成绩对此次比赛的胜负做个大致的判断。后验分布一般也认为是在给定样本的情况下的条件分布。

综合起来看，则好比是人类刚开始时对大自然只有少得可怜的先验知识，但随着不断的观察、实验获得更多的样本、结果，使得人们对自然界的规律摸得越来越透彻。所以，贝叶斯方法既符合人们日常生活的思考方式，也符合人们认识自然的规律，经过不断的发展，最终占据统计学领域的半壁江山，与经典统计学分庭抗礼。

2 基本概念

在引出贝叶斯定理之前，先介绍几个定义：

* 条件概率就是事件在另外一个事件已经发生条件下的发生概率。条件概率表示为，读作“在条件下的概率”。
* 联合概率表示两个事件共同发生的概率。与的联合概率表示为或者。
* 边缘概率（又称先验概率）是某个事件发生的概率。边缘概率是这样得到的：在联合概率中，把最终结果中那些不需要的事件通过合并成它们的全概率，而消去它们（对离散随机变量用求和得全概率，对连续随机变量用积分得全概率），这称为边缘化（marginalization），比如的边缘概率表示为，的边缘概率表示为。

接着，考虑一个问题：是在发生的情况下发生的可能性。

1. 首先，事件发生之前，我们对事件的发生有一个基本的概率判断，称为的先验概率，用表示；
2. 其次，事件发生之后，我们对事件的发生概率重新评估，称为的后验概率，用表示；
3. 类似的，事件发生之前，我们对事件的发生有一个基本的概率判断，称为的先验概率，用表示；
4. 同样，事件发生之后，我们对事件的发生概率重新评估，称为的后验概率，用表示；

3 数学推导

贝叶斯定理便是基于下述贝叶斯公式：

具体推导过程如下：

根据条件概率的定义，在事件发生的条件下事件发生的概率是：

同样的，在事件发生的条件下事件B发生的概率是：

将两者整理与合并，我们就可以得到：

4 定理理解

可能许多人在看完上述理论推导以后，还是不能很好理解贝叶斯定理，下面我就以一个例子来说明贝叶斯定理的应用。

例1：假设有两个各装了100个球的箱子，甲箱子中有70个红球，30个绿球，乙箱子中有30个红球，70个绿球。假设随机选择其中一个箱子，从中拿出一个球记下球色再放回原箱子，如此重复12次，记录得到8次红球，4次绿球。问题来了，你认为被选择的箱子是甲箱子的概率有多大？

《决策与判断》这本书中针对这个问题做了问卷调查，结果大多数人都低估了选择是甲盒子的概率。事实上，根据贝叶斯定理，正确答案是96.7%。下面我来进行解释：

刚开始选择甲乙两箱子的先验概率都是50%，因为是随机二选一（这是贝叶斯定理二选一的特殊形式）。即有：

P(甲) = 0.5， P(乙) = 1 - P(甲)；

这时在拿出一个球是红球的情况下，我们就应该根据这个信息来更新选择的是甲箱子的先验概率：

P(甲|红球1) = P(红球|甲) × P(甲) / (P(红球|甲) × P(甲) + (P(红球|乙) × P(乙)))

P(红球|甲)：甲箱子中拿到红球的概率

P(红球|乙)：乙箱子中拿到红球的概率

因此在出现一个红球的情况下，选择的是甲箱子的先验概率就可被修正为：

P(甲|红球1) = 0.7 × 0.5 / (0.7 × 0.5 + 0.3 × 0.5) = 0.7

即在出现一个红球之后，甲乙箱子被选中的先验概率就被修正为：

P(甲) = 0.7， P(乙) = 1 - P(甲) = 0.3；

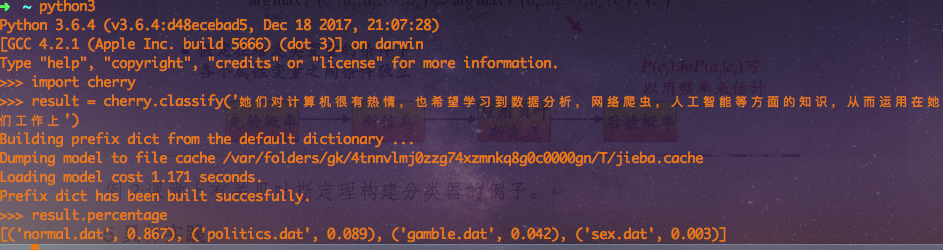
如此重复，直到经历8次红球修正（概率增加），4此绿球修正（概率减少）之后，选择的是甲箱子的概率为：96.7%。

5 具体实践

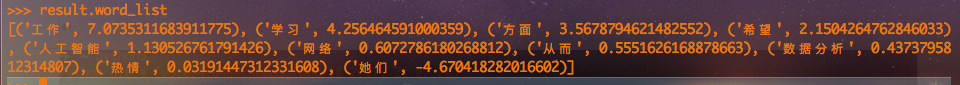
贝叶斯模型在机器学习以及人工智能中都有出现，cherry 分类器使用了朴素贝叶斯模型算法，经过简单的优化，使用 1000 个训练数据就能得到 97.5% 的准确率。

使用pip安装cherry，pip install cherry；

终端下使用cherry进行实验：

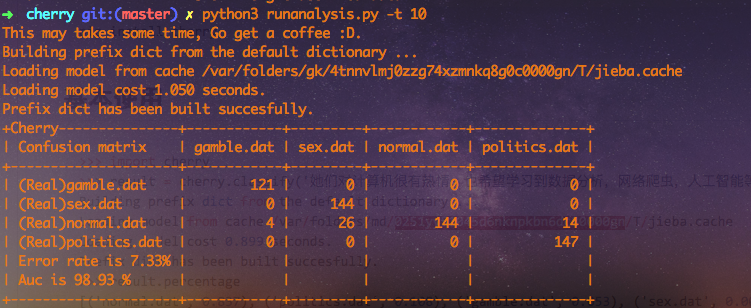


上述例子展示了输入的句子数据对应每个类别的概率，其中正常句子的概率为86.7%，政治敏感概率为8.9%，赌博为4.2%，色情概率为0.3%。



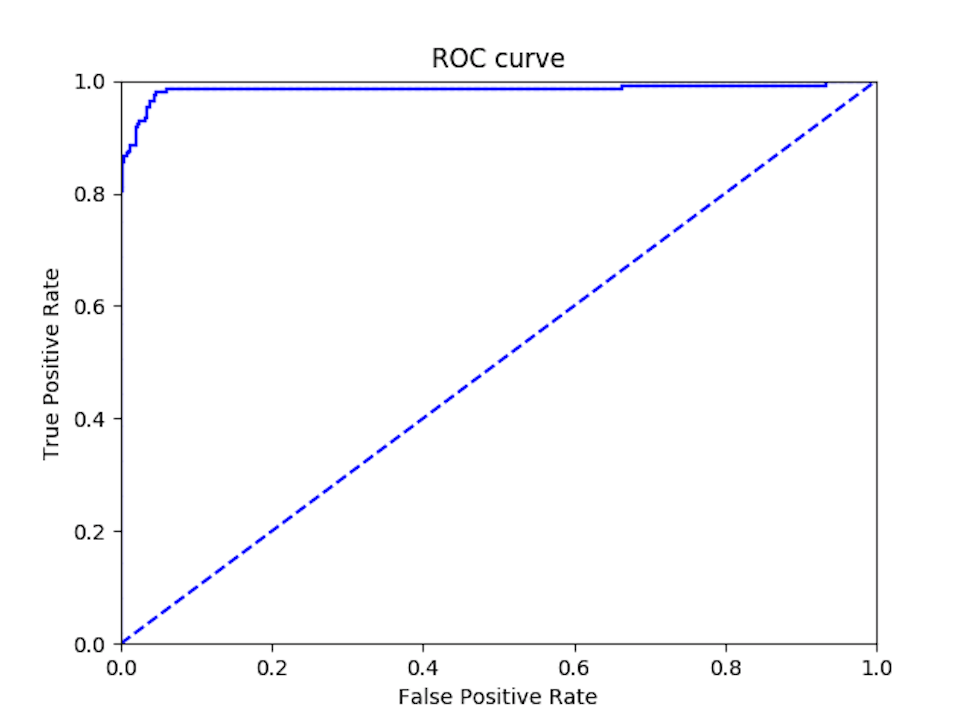
result的word\_list属性显示的是句子的有效部分，上面例子中，句子被正确分类成正常类别，影响度最大的词语分别是“工作”、“学习”、“方面”；

我们在网上通过下载测试数据，对测试脚本进行测试；测试脚本默认从中文数据中随机选取60个数据做为测试数据，剩下的数据用作训练数据，重复10次；



得到混淆矩阵以及准确率，如上；

其中查准率我们看到为100%，查全率为144/(4+26+144+14) = 76.5%

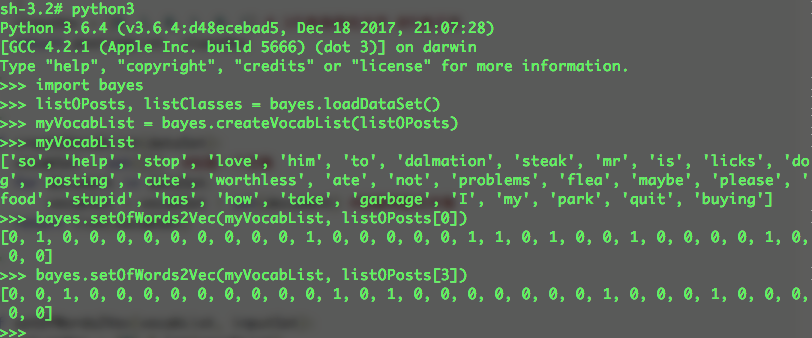
我们计算得到的AUC为：

接下来我们通过自己编写一个训练器，从而来更深入理解贝叶斯模型。

**我们以在线社区留言板为例，通过对留言内容进行过滤，识别出侮辱类和非侮辱类言论，使用1和0分别标识：**

**我们把文本转换成向量，考虑出现在所有文档中的所有单词，必须要将每一篇文档转换为词汇表上的向量，下面为从词表到向量的转换函数：**

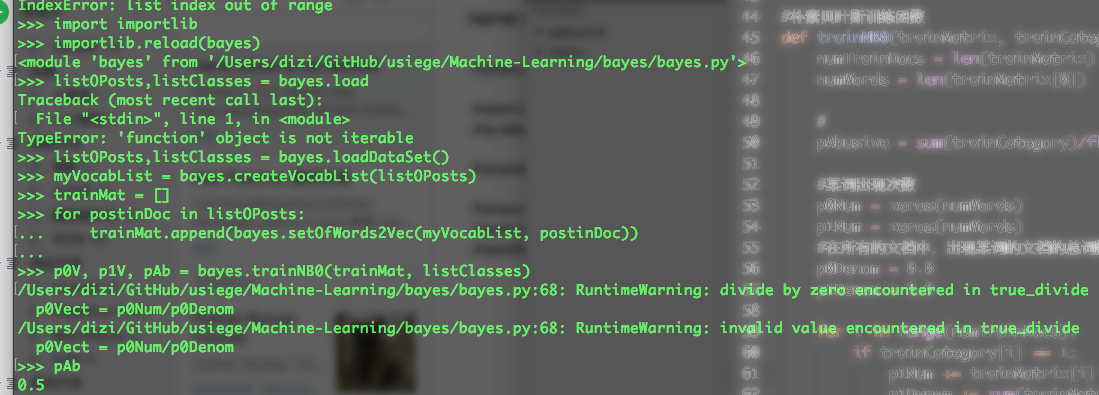
1. 从文本中创建词向量bayes.py
2. #!usr/bin/python
3. #-\*-encoding:utf-8-\*-
4. '''
5. 该函数返回实验样本，该样本被切分成词条集合；
6. 第二个变量返回类别，该类别由人工标注，用于训练程序以便自动检查侮辱性留言；
7. '''
8. def loadDataSet():
9. postingList = [
10. ['my','dog','has','flea','problems','help','please'],
11. ['maybe','not','take','him','to','dog','park','stupid'],
12. ['my','dalmation','is','so','cute','I','love','him'],
13. ['stop','posting','stupid','worthless','garbage'],
14. ['mr','licks','ate','my','steak','how','to','stop','him'],
15. ['quit','buying','worthless','dog','food','stupid']
16. ]
17. classVec = [0, 1, 0, 1, 0, 1] # 1代表侮辱性文字 0代表正常
18. return postingList, classVec
19. '''
20. '''
21. def createVocabList(dataSet):
22. vocabSet = set([]) #创建一个空集
23. for document in dataSet:
24. vocabSet = vocabSet | set(document) #创建两集合并集
25. return list(vocabSet)
26. '''
27. 该函数输入参数为词汇表及某个文档，输出的是文档向量，向量每一元素为1or0，分别表示词汇表中的单词在输入文档中是否出现
28. '''
29. def setOfWords2Vec(vocabList, inputSet):
30. returnVec = [0] \* len(vocabList)
31. for word in inputSet:
32. if word in vocabList:
33. returnVec[vocabList.index(word)] = 1
34. else:
35. print("the word: %s is not in my Vocabulary!" % word)
36. return returnVec

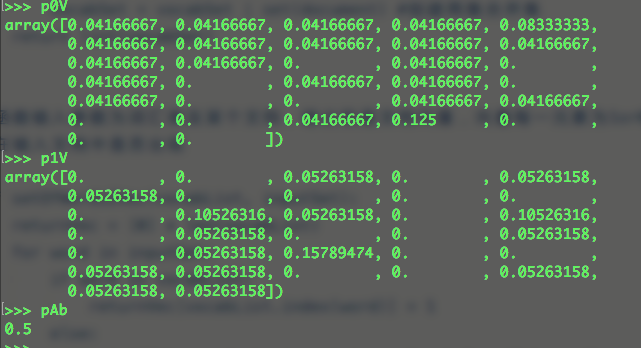


函数loadDataSet创建了一个实验样本，该样本是进行词条切分后的文档集合，而后一变量是通过人工标注的类别集合，用于训练程序。

2. 朴素贝叶斯训练函数

1. #朴素贝叶斯训练函数
2. def trainNB0(trainMatrix, trainCategory):
3. numTrainDocs = len(trainMatrix)
4. numWords = len(trainMatrix[0])
5. #
6. pAbusive = sum(trainCategory)/float(numTrainDocs)
7. #某词出现次数
8. p0Num = zeros(numWords)
9. p1Num = zeros(numWords)
10. #在所有的文档中，出现某词的文档的总词数
11. p0Denom = 0.0
12. p1Denom = 0.0
13. for i in range(numTrainDocs):
14. if trainCategory[i] == 1:
15. p1Num += trainMatrix[i]
16. p1Denom += sum(trainMatrix[i])
17. else:
18. p0Num += trainMatrix[i]
19. p0Denom += sum(trainMatrix[i])
20. p1Vect = p1Num/p1Denom
21. p0Vect = p0Num/p0Denom
22. return p0Vect, p1Vect, pAbusive

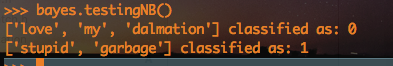


   
3. 修改分类器   
\* Problem1:计算多个概率的乘积以获得文档属于某个类别概率，如果其中有一个概率值为0，那最后乘积也为0；为降低这种影响，可以将所有词出现初始化为1，并将分母初始化为2

1. p0Num = ones(numWords);
2. p1Num = ones(numWords)
3. p0Denom = 2.0;
4. p1Denom = 2.0

* Problem2: 下溢出，太多很小的数相乘会造成下溢出，解决办法是取自然对数，把乘法转换成加法，通过求对数避免下溢出或者浮点数舍入导致错误

1. p1Vect = log(p1Num/p1Denom)
2. p0Vect = log(p0Num/p0Denom)
3. 分类器编写
4. #构建朴素贝叶斯分类函数
5. def classityNB(vec2Classify, p0Vec, p1Vec, pClass1):
6. p1 = sum(vec2Classify \* p1Vec) + log(pClass1)
7. p0 = sum(vec2Classify \* p0Vec) + log(1.0 - pClass1)
8. if p1 > p0:
9. return 1;
10. else:
11. return 0;
12. def testingNB():
13. listOPosts, listClasses = loadDataSet()
14. myVocabList = createVocabList(listOPosts)
15. trainMat = []
16. for postinDoc in listOPosts:
17. trainMat.append(setOfWords2Vec(myVocabList, postinDoc))
18. p0V, p1V, pAb = trainNB0(array(trainMat), array(listClasses))
19. testEntry = ['love', 'my', 'dalmation']
20. thisDoc = array(setOfWords2Vec(myVocabList, testEntry))
21. print(testEntry, 'classified as:', classityNB(thisDoc, p0V, p1V, pAb))
22. testEntry = ['stupid', 'garbage']
23. thisDoc = array(setOfWords2Vec(myVocabList, testEntry))
24. print(testEntry, 'classified as:', classityNB(thisDoc, p0V, p1V, pAb))

通过训练器分类得出结果：   


6 应用领域

6.1 拼写检查

经常在网上搜索东西的朋友知道，当你不小心输入一个不存在的单词时，搜索引擎会提示你是不是要输入某一个正确的单词，比如当你在Google中输入“kabe bryant”时，系统会提示你是不是要搜索“kobe bryant”；

下面我们来看看这种拼写检查是如何实现的：

用户输入一个单词时，可能拼写正确，也可能拼写错误。如果把拼写正确的情况记作（代表correct），拼写错误的情况记作（代表wrong），那么拼写检查要做的事情就是：在发生的情况下，试图推断出。换言之，已知，然后在若干个备选方案中，找出可能性最大的那个，也就是求的最大值。

而根据贝叶斯定理，有

由于对于所有备选的来说，对应的都是同一个，所以它们的是相同的，因此我们只要最大化：

即可。其中：

表示某个正确的词的出现"概率"，它可以用"频率"代替。如果我们有一个足够大的文本库，那么这个文本库中每个单词的出现频率，就相当于它的发生概率。某个词的出现频率越高，就越大。

表示在试图拼写的情况下，出现拼写错误的概率。为了简化问题，假定两个单词在字形上越接近，就越有可能拼错，就越大。举例来说，相差一个字母的拼法，就比相差两个字母的拼法，发生概率更高。你想拼写单词Kobe，那么错误拼成Kabe（相差一个字母）的可能性，就比拼成Kaabe高（相差两个字母）。

所以，我们只要找到与输入单词在字形上最相近的那些词，再在其中挑出出现频率最高的一个，就能实现的最大值。

6.2医患诊断模型

7岁女孩晓宇（化名）患急性支气管炎,在武汉市儿童医院住院4天，医生为确诊病情，为她抽血化验了32个指标，仅化验费就花费1130元。晓宇的家长质疑：医院如此看病，是过度检查。晓宇的接诊医生李志超说：“晓宇入院时,根据其家长自述病情，我认为孩子的情况有些严重,于是确定了上述化验指标”。该院四内科副主任李医生说：在当时情况下,李志超对患者的病情判断、以及开出的化验指标,都是有道理的。但如果是我接诊，会以自己的经验有针对性地进行化验检查,可能不会一下开出这么多化验指标。该科主任温玟莉主任医师称：一次抽血化验32个指标，是因为李志超当时怀疑孩子得了败血症，这样处理没有问题。但最后的检查结果并不是败血症，这只能说明李志超较年轻，缺乏丰富的临床经验，只有通过全面检查才能确诊。

在医患关系紧张，看病难、看病贵的现实情况下，我们应如何看待这个颇有争议的案例，医生看病是应该有针对性地开方,还是列出“算法式”的化验指标进行排查，本研究以贝叶斯公式为依据，从中国现行的医疗体制出发,对此类问题进行了有益的探索，以期建立一种定量化的诊断模型。

模型建立：模型建立 设“患者有某种病症”为事件A,引起事件A的病因为样本空间Ω。B1,B2,…Bn为Ω的一个分划,即Bi∩Bj=Φ,i≠ j,Un i=1Bi=Ω,并假定P(Bi)>0。由贝叶斯公式,由某病因引 起事件A的概率为:

P(Bi|A)= P(Bi)P(A|Bi) 6n j=1 P(Bj)P(A⌒Bj) (1)

公式(1)为医生有针对性地确诊提供了参考。 在疹疗过程中,医生要根据临床经验对各种病因Bi进行权衡。如果误诊,则有可能承担相应的医疗事故风险,相应的 误诊概率记为P′ (Bi),并设因可能承担风险而承担的赔偿费用为C′i,患者承担医生针对病因Bi开出的疹疗方案的费用为 Ci,于是在一次诊治过程中患者承担的平均费用为: E(A)=6n i=1P(Bi)Ci (2) 医生可能承担的平均赔偿金额为: E′(A)=6n i=1P′(Bi)C′i (3) 我们称该模型为诊断模型,并以δ1≤E(A)-E′(A)≤δ2 为标准来衡量诊断方案的合理性,其中δ1≥0,δ2为某一不是特别大的正数。即患者所承担的平均医疗费用应比医生可能承担的平均赔偿金要多,但两者不应差别太大。

6.3垃圾邮件过滤器

垃圾邮件是一种令人头痛的顽症，困扰着所有的互联网用户。

正确识别垃圾邮件的技术难度非常大。传统的垃圾邮件过滤方法，主要有"关键词法"和"校验码法"等。前者的过滤依据是特定的词语；后者则是计算邮件文本的校验码，再与已知的垃圾邮件进行对比。它们的识别效果都不理想，而且很容易规避。

2002年，Paul Graham提出使用"贝叶斯推断"过滤垃圾邮件。他说，这样做的效果，好得不可思议。1000封垃圾邮件可以过滤掉995封，且没有一个误判。

另外，这种过滤器还具有自我学习的功能，会根据新收到的邮件，不断调整。收到的垃圾邮件越多，它的准确率就越高。

贝叶斯过滤器是一种统计学过滤器，建立在已有的统计结果之上。所以，我们必须预先提供两组已经识别好的邮件，一组是正常邮件，另一组是垃圾邮件。

我们用这两组邮件，对过滤器进行"训练"。这两组邮件的规模越大，训练效果就越好。Paul Graham使用的邮件规模，是正常邮件和垃圾邮件各4000封。

"训练"过程很简单。首先，解析所有邮件，提取每一个词。然后，计算每个词语在正常邮件和垃圾邮件中的出现频率。比如，我们假定"sex"这个词，在4000封垃圾邮件中，有200封包含这个词，那么它的出现频率就是5%；而在4000封正常邮件中，只有2封包含这个词，那么出现频率就是0.05%。（如果某个词只出现在垃圾邮件中，Paul Graham就假定，它在正常邮件的出现频率是1%，反之亦然。这样做是为了避免概率为0。随着邮件数量的增加，计算结果会自动调整。）

有了这个初步的统计结果，过滤器就可以投入使用了。

7 参考文献

1.An Essay towards solving a problem in the Doctrine of chances. <http://rstl.royalsocietypublishing.org/content/53/370.full.pdf+html>

2.[美]Peter Harrington.Machine Learning in Action[M]. April 2012

1. 《An essay towards solving a problem in the doctrine of chances》 [↑](#footnote-ref-2)