朴素贝叶斯之原理和实现

汪加林

2016 年 12 月 28 日

1 简介

这一节主要是用来介绍下朴素贝叶斯的原理和实现的。朴素贝叶斯的原理非常简单，实现起来也非常简单。

本文档的内容结构如下：

- 原理
- 实现
- 类的设计

2 朴素贝叶斯的原理

2.1 问题提出

朴素贝叶斯通常用来分类，例如文本分类达到过滤垃圾邮件，情感分析的目的。看下面两句话：

1. stop posting stupid worthless garbage
2. my dog is so cute, i love him

第一句带有否定、责怪的语气，这是由 stupid（愚蠢）体现的；第二句则是高兴、开心的语气，这是由 love （喜欢）、cute（可爱）体现的。

朴素贝叶斯分类器就通过在文本中出现的单词来分类，分类器将文本中出现的单词视为特征，如果文本中出现很多积极的词语，那么这句话情感极性被判断为正的可能性就越大；反之如果出现很多否定、消极的词语，则情感极性被判断为负的可能性就越大。

2.2 理论推导

朴素贝叶斯的原理非常简单，基于贝叶斯定理，冠以“朴素”两字，是因为它有下面的这个假设：

- 用于分类的特征相互之间都是独立的。

当然这个假设大多数时候并不是严格成立的。以上文中例子来说，当文本中出现一堆词 $x = (x_1, x_2, ..., x_n)$ 之后，如何判断它是属于哪一类 $(y_k \in -1, 1)$ 呢？

$$
\begin{aligned}
p(y = y_k | x = (x_1, x_2, ..., x_n)) &= \frac{p(y = y_k, x = (x_1, x_2, ..., x_n))}{p(x = (x_1, x_2, ..., x_n))} && (1)\\
&= \frac{p(x = (x_1, x_2, ..., x_n) | y = y_k) * p(y = y_k)}{p(x = (x_1, x_2, ..., x_n))} && (2)\\
&= \frac{\prod_{i=1}^{n} p(x_i | y = y_k) * p(y = y_k)}{p(x = (x_1, x_2, ..., x_n))} && (3)\\
& && (4)
\end{aligned}
$$

上式中，$p(y_k), p(x_i|y = y_k)$ 都可以在训练的时候计算出来，$p(y_k)$ 就是类分布的先验概率，而 $p(x_i|y = y_k)$ 就是在不同类别中每个特征出现的概率。最后，文本的类别是后验概率 $p(y = y_k|x = (x_1, x_2, ..., x_n))$ 最大的那一类，即：

$$y = argmax_{y_k} p(y = y_k | x = (x_1, x_2, ..., x_n))$$

2.3 概率的计算

计算 $p(y_k)$ 非常简单，麻烦点的便是 $p(y = y_k|x = (x_1, x_2, ..., x_n))$ 的计算了。实际中，特征的分布可能不同，因此有不同的计算方式。在sklearn中实现了高斯分布、多项式分布、伯努利分布几种不同的方式。在这里，我们还是以文本为例，只介绍假设特征值满足多项式分布时，如何计算 $p(y = y_k|x = (x_1, x_2, ..., x_n))$。

设在不同类别 y 下，特征的分布为 $\theta_y = (\theta_{y1}, \theta_{y2}, ..., \theta_{yn})$，$n$ 是所有特征数（在文本分类中，就是单词的个数）。因此 $p(x_i|y) = \theta_{y_i}$。

$$\theta_{yi} = \frac{N_{xi} + \alpha}{N_y + n * \alpha}$$

其中，N_{yi} 是在类别 y 中第 i 个特征（词）出现的次数，N_y 是类别 y 中所有特征（词）出现的次数总和。α 是拉普拉斯平滑参数，防止 $N_{yi} = 0$，概率为 0 的情况。

3 算法实现

上面讲了朴素贝叶斯算法的整个原理和数学推导。但是实现往往会有点不同，每一个好的实现都是数学与工程的折衷。同样的，在实现朴素贝叶斯算法时，下面是一些我觉得要注意的地方。

3.1 实现时要注意的地方

1. 计算后验概率时的分母 $p(x=(x_1,x_2,...,x_n))$, 它只是起归一化作用，我们只需要选择分子最大的那一类即可；
2. 计算 $\prod_{i=1}^{n} p(x_i|y=y_k)$ 时，由于是连乘，当特征很多，而且概率都很小的时候，连乘有可能做成溢出，结果错误；因此我们对所有的概率先取对数，将连乘变成加法；
3. 如果还想得到分类的概率大小，那么就将对数概率进行指数运算，同时进行求和归一化即可；
4. 在计算 $p(x_i|y=y_k)$ 时，由于实际的应用中数据的分布不同有不同的计算方式，例如高斯分布、多项式分布、伯努利分布；
5. 由于有可能出现 $p(x_i|y=y_k)$ 为 0 的情况，因此实际中会使用拉普拉斯平滑，将所有的计数都加上一个常数, 如之前计算概率那一节内容所示。

3.2 算法实现

Algorithm 1 Naive Bayes

compute $p(y=y_k)$
compute $p(x_i|y=y_k)$
for each y_k **do**
  $p(y=y_k)=0$
  for each x_i **do**
    $p(y=y_k)=p(y=y_k)+log(p(x_i|y=y_k))$
  end for
end for
$y=argmax_{y_k}p(y=y_k)$

4 类的设计

- data_utils: 用来处理文本数据的类，将文本映射成整数
- naive_bayes：实现朴素贝叶斯算法的主类