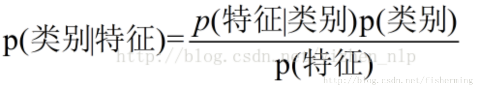
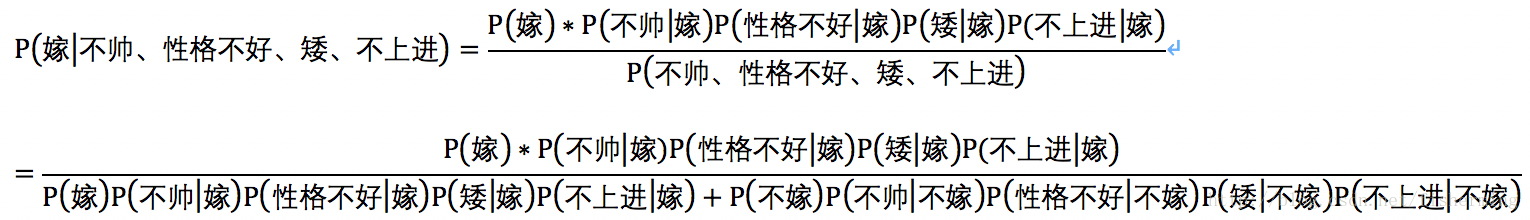
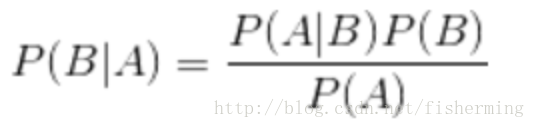
朴素贝叶斯

  朴素贝叶斯中的朴素一词的来源就是假设各特征之间相互独立。这一假设使得朴素贝叶斯算法变得简单，但有时会牺牲一定的分类准确率。对于目标求解为不同的类别，贝叶斯公式的分母总是相同的。所以，只求解分子即可。对于分母，条件独立性假设是假定“输入参数的各个特征”对“输出参数的值”的影响相互独立，而不是假设特征本身相互独立。 所以“P(不帅、性格不好、矮、不上进)”并不等于“P(不帅)P(性格不好)P(矮)P(不上进)”。





优点：

算法逻辑简单,易于实现(算法思路很简单，只要使用贝叶斯公式转化即可）

缺点：

朴素贝叶斯假设属性之间相互独立，这种假设在实际过程中往往是不成立的。在属性之间相关性越大，分类误差也就越大。

GloVe

设共现矩阵为X，其元素为Xi,j。X i,j的意义为：在整个语料库中，单词i和单词j共同出现在一个窗口中的次数。(如5 love but you love him i)。GloVe模型没有使用神经网络的方法。

Glove和skip-gram、CBOW模型对比

我的理解是skip-gram、CBOW每次都是用一个窗口中的信息更新出词向量，但是Glove则是用了全局的信息（共线矩阵），也就是多个窗口进行更新

Keras

在Keras中有两类主要的模型：Sequential顺序模型和使用函数式API的Model 类模型。

Keras的底层库使用Theano或TensorFlow，这两个库也称为Keras的后端。无论是Theano还是TensorFlow，都是一个“符号式”的库。

因此，这也使得Keras的编程与传统的Python代码有所差别。笼统的说，符号主义的计算首先定义各种变量，然后建立一个“计算图”，计算图规定了各个变量之间的计算关系。建立好的计算图需要编译以确定其内部细节，然而，此时的计算图还是一个“空壳子”，里面没有任何实际的数据，只有当你把需要运算的输入放进去后，才能在整个模型中形成数据流，从而形成输出值。

就像用管道搭建供水系统，当你在拼水管的时候，里面是没有水的。只有所有的管子都接完了，才能送水。

Keras的模型搭建形式就是这种方法，在你搭建Keras模型完毕后，你的模型就是一个空壳子，只有实际生成可调用的函数后（K.function），输入数据，才会形成真正的数据流。

深度学习的优化算法，说白了就是梯度下降。每次的参数更新有两种方式。

第一种，遍历全部数据集算一次损失函数，然后算函数对各个参数的梯度，更新梯度。这种方法每更新一次参数都要把数据集里的所有样本都看一遍，计算量开销大，计算速度慢，不支持在线学习，这称为Batch gradient descent，批梯度下降。

另一种，每看一个数据就算一下损失函数，然后求梯度更新参数，这个称为随机梯度下降，stochastic gradient descent。这个方法速度比较快，但是收敛性能不太好，可能在最优点附近晃来晃去，hit不到最优点。两次参数的更新也有可能互相抵消掉，造成目标函数震荡的比较剧烈。

为了克服两种方法的缺点，现在一般采用的是一种折中手段，mini-batch gradient decent，小批的梯度下降，这种方法把数据分为若干个批，按批来更新参数，这样，一个批中的一组数据共同决定了本次梯度的方向，下降起来就不容易跑偏，减少了随机性。另一方面因为批的样本数与整个数据集相比小了很多，计算量也不是很大。