朴素贝叶斯（Naïve Bayes）

介绍

Byesian算法是统计学的分类方法，它是一种利用概率统计知识进行分类的算法。在许多场合，朴素贝叶斯分类算法可以与决策树和神经网络分类算法想媲美，该算法能运用到大型数据库中，且方法简单，分类准确率高，速度快，这个算法是从贝叶斯定理的基础上发展而来的，贝叶斯定理假设不同属性值之间是不相关联的。但是现实说中的很多时候，这种假设是不成立的，从而导致该算法的准确性会有所下降。

运用场景

1.医生对病人进行诊断就是一个典型的分类过程，任何一个医生都无法直接看到病人的病情，只能观察病人表现出的症状和各种化验检测数据来推断病情，这时医生就好比一个分类器，而这个医生诊断的准确率，与他当初受到的教育方式（构造方法）、病人的症状是否突出（待分类数据的特性）以及医生的经验多少（训练样本数量）都有密切关系。

2.根据各种天气状况判断一个人是否会去踢球，下面的例子就是。

3.各种分类场景

贝叶斯定理

已知某条件概率，如何得到两个事件交换后的概率，也就是在已知P(A|B)的情况下如何求得P(B|A)。这里先解释什么是条件概率：

表示事件B已经发生的前提下，事件A发生的概率，叫做事件B发生下事件A的条件概率。其基本求解公式为：$$P(A|B) = \frac{P(AB)}{P(B)}$$

贝叶斯定理之所以有用，是因为我们在生活中经常遇到这种情况：我们可以很容易直接得出P(A|B)，P(B|A)则很难直接得出，但我们更关心P(B|A)，贝叶斯定理就为我们打通从P(A|B)获得P(B|A)的道路。

下面直接给出贝叶斯定理：

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

输入数据说明

数据:天气情况和每天是否踢足球的记录表

日期	踢足球	天气	温度	湿度	风速
1号	否(0)	晴天(0)	热(0)	高(0)	低(0)
2号	否(0)	晴天(0)	热(0)	高(0)	高(1)
3号	是(1)	多云(1)	热(0)	高(0)	低(0)

4号	是(1)	下雨(2)	舒适(1)	高(0)	低(0)
5号	是(1)	下雨(2)	凉爽(2)	正常(1)	低(0)
6号	否(0)	下雨(2)	凉爽(2)	正常(1)	高(1)
7号	是(1)	多云(1)	凉爽(2)	正常(1)	高(1)
8号	否(0)	晴天(0)	舒适(1)	高(0)	低(0)
9号	是(1)	晴天(0)	凉爽(2)	正常(1)	低(0)
10号	是(1)	下雨(2)	舒适(1)	正常(1)	低(0)
11号	是(1)	晴天(0)	舒适(1)	正常(1)	高(1)
12号	是(1)	多云(1)	舒适(1)	高(0)	高(1)
13号	是(1)	多云(1)	热(0)	正常(1)	低(0)
14号	否(0)	下雨(2)	舒适(1)	高(0)	高(1)
15号	?	晴天(0)	凉爽(2)	高(0)	高(1)

数据抽象为如下，含义为是否会去踢球，天气，温度，湿度，风速

```
0,0 0 0 0
0,0 0 0 1
1,1 0 0 0
1,2 1 0 0
1,2 2 1 0
0,2 2 1 1
1,1 2 1 1
0,0 1 0 0
1,0 2 1 0
1,2 1 1 0
1,0 1 1 1
1,1 1 0 1
1,1 0 1 0
0,2 1 0 1
```

如果15号的天气为(晴天，凉爽，湿度高，风速高，预测他是否会踢足球)

计算过程

假设小明15号去踢球，踢球概率为：

P(踢)=9/14

P(晴天|踢)=2/9

P(凉爽|踢)=3/9

P(湿度高|踢)=3/9

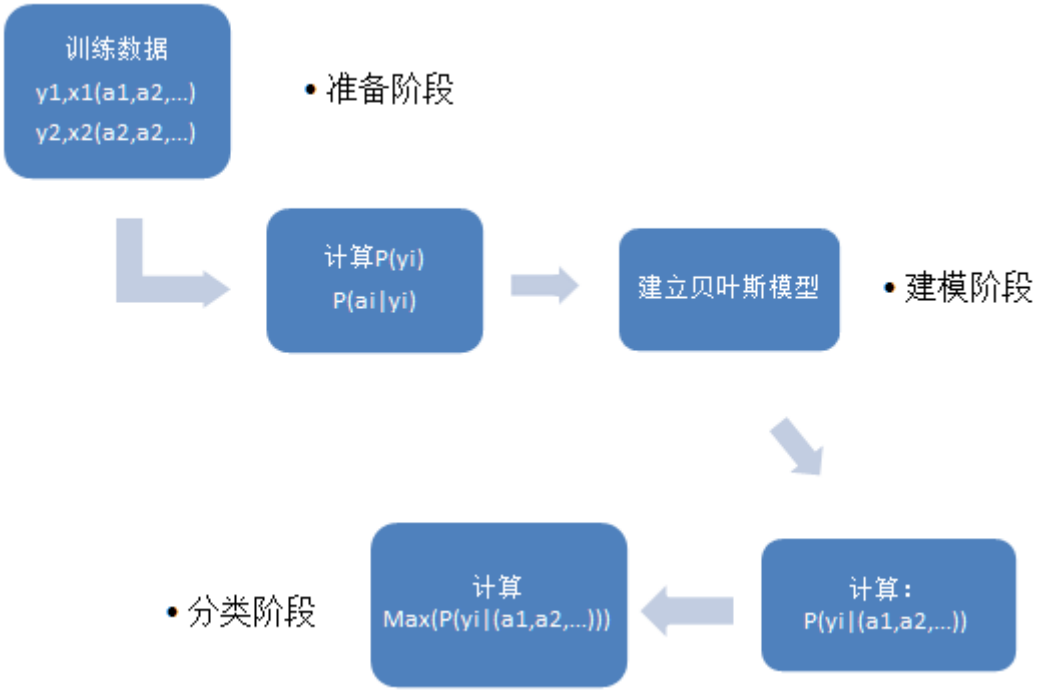
P(风速高|踢)=3/9

P(踢)由踢的天数除以总天数得到，P(晴天|踢)为踢球的同事是晴天除以踢的天数得到，其他以此类推。
P(踢|晴天,凉爽,湿度高,风速高)=
P(踢)* P(晴天|踢)* P(凉爽|踢)* P(湿度高|踢) *P(风速高|踢)=
9/14*2/9*3/9*3/9*3/9=0.00529

假设小明15号不去踢球，概率为：
P(不踢)=5/14
P(晴天|不踢)=3/5
P(凉爽|不踢)=1/5
P(湿度高|不踢)=4/5
P(风速高|不踢)=3/5
P(不踢|晴天,凉爽,湿度高,风速高)=
P(不踢)* P(晴天|不踢)* P(凉爽|不踢)* P(湿度高|不踢) *P(风速高|不踢)=
5/14*3/5*1/5*4/5*3/5=0.02057

可以看到小明不去踢足球的概率比去踢足球的概率高。

流程图



测试代码

```
importorg.apache.spark.mllib.classification.NaiveBayes
import org.apache.spark.mllib.linalg.Vectors
import org.apache.spark.mllib.regression.LabeledPoint
import org.apache.spark.{SparkContext,SparkConf}

object naiveBayes {
  def main(args: Array[String]) {
    val conf =new SparkConf()
    val sc =new SparkContext(conf)

    //读入数据
    val data = sc.textFile(args(0))
    val parsedData =data.map { line =>
```

```
      val parts =line.split(',')
      LabeledPoint(parts(0).toDouble,Vectors.dense(parts(1).split(' ').map(_.toDouble)))
    }
    // 把数据的60%作为训练集，40%作为测试集.
    val splits = parsedData.randomSplit(Array(0.6,0.4),seed = 11L)
    val training =splits(0)
    val test =splits(1)

    //获得训练模型,第一个参数为数据，第二个参数为平滑参数，默认为1，可改
    val model =NaiveBayes.train(training,lambda = 1.0)

    //对模型进行准确度分析
    val predictionAndLabel= test.map(p => (model.predict(p.features),p.label))
    val accuracy =1.0 *predictionAndLabel.filter(x => x._1 == x._2).count() / test.count()

    println("accuracy-->"+accuracy)
    println("Predictionof (0.0, 2.0, 0.0, 1.0):"+model.predict(Vectors.dense(0.0,2.0,0.0,1.0)))
  }
}
```

提交代码脚本(standalone模式)：

```
./bin/spark-submit
--name nb                          （项目名）
--class naiveBayes                 （主类名）
--master spark://master:7077       （使用集群管理器）
~/Desktop/naiveBayes.jar           （代码包位置）
Hdfs://master:9000/NB.data         （args(0)的参数值）
```

输出结果说明

```
accuracy-->0.75
```

准确度为75%,这里是因为测试集数据量比较小的原因，所以偏差较大。

```
Prediction of (0.0, 2.0, 0.0, 1.0):0.0
```

可以从结果看到对15号的预测为不会踢球，和我们数学计算的结果一致。