基于 MR 的朴素贝叶斯分类器设计与实现

江穹峰 2017.03

华中科技大水电与数字化工程学院信息与控制研究所 湖北 武汉 430074

摘要：

贝叶斯分类是一系列分类算法的总称，这类算法均以贝叶斯定理为基础，故统称为贝叶斯分类。朴素贝叶斯分类器是一系列以假设特征之间强（朴素）独立下运用贝叶斯定理为基础的简单概率分类器。Map/Reduce 是 Google 提出的一个软件架构，用于大规模数据集的并行运算。概念“Map（映射）”和“Reduce（归纳）”，及他们的主要思想，都是从函数式编程语言借来的，还有从矢量编程语言借来的特性。当前的软件实现是指定一个 Map（映射）函数，用来把一组键值对映射成一组新的键值对，指定并发的 Reduce（归纳）函数，用来保证所有映射的键值对中的每一个共享相同的键组。

关键字：概率分类器、贝叶斯定理、Map/Reduce 编程模型

1. 朴素贝叶斯分类器理论介绍

朴素贝叶斯是一种构建分类器的简单方法。该分类器模型会给问题实例分配用特征值表示的类标签，类标签取自有限集合。它不是训练这种分类器的单一算法，而是一系列基于相同原理的算法：所有朴素贝叶斯分类器都假定样本每个特征与其他特征都不相关。举个例子，如果一种水果其具有红，圆，直径大概 3 英寸等特征，该水果可以被判定为是苹果。尽管这些特征相互依赖或者有些特征由其他特征决定，然而朴素贝叶斯分类器认为这些属性在判定该水果是否为苹果的概率分布上独立的。对于某些类型的概率模型，在监督式学习的样本集中能获取得非常好的分类效果。在许多实际应用中，朴素贝叶斯模型参数估计使用最大似然估计方法；换而言之，在不用到贝叶斯概率或者任何贝叶斯模型的情况下，朴素贝叶斯模型也能奏效。

朴素贝叶斯方法（Naïve Bayes）

问题：给定一个类标签集合 C = {c1,c2, …, cj}以及一个文档 d，给文档 d 分配一个最合适的类别标签。

基本思想：对于类标签集合 C 中的每个类标签 ci (i = 1, …, j), 计算条件概率 p (ci | d)，使条件概率 p (ci | d)最大的类别作为文档 d 最终的类别。基于概率的思想，给定一个文档 d，看该文档属于哪个类别的可能性最大，就认为该文档属于哪个类别。因此 Naïve Bayes 是一个基于概率的分类器。

贝叶斯公式 $$p(c_i \mid d) = \frac{p(c_i, d)}{p(d)} = \frac{p(d \mid c_i)\, p(c_i)}{p(d)}$$

即 $$posterior = \frac{likelihood \times prior}{evidence}$$

p (ci | d) : 后验概率或条件概率(posterior)　　p (ci): 先验概率(prior)

p (d | ci): 似然概率(likelihood)　　p (d): 证据(evidence)

Bayes 公式的意义：当观察到 evidence p (d) 时，后验概率 p (ci | d)取决于似然概率 p (d | ci)和先验概率 p (ci)。因为当 evidence p (d) 已知时，p (d)成为常量，Bayes 公式变成

$$p(c_i \mid d) = \frac{p(d \mid c_i)\, p(c_i)}{p(d)} \propto p(d \mid c_i)\, p(c_i)$$

对于类标签集合 C 中的每个类标签 ci (i = 1, …, j), 计算条件概率 p (ci |d)，使条件概率 p (ci | d)最大的类别作为文档 d 最终的类别。

$$c_d = \arg\max_{c_i \in \mathbf{C}} p(c_i \mid d) = \arg\max_{c_i \in \mathbf{C}} p(d \mid c_i)\, p(c_i)$$

其中 Cd 为文档 d 被赋予的类型，Cd=使得条件概率 p (ci |d)最大的类型。根据 Bayes 公式，Cd=使得 p (d | ci) p (ci)值最大的类型。

对于 Naïve Bayes，训练集就是用来计算 p (d | ci) 和 p (ci)，方法如下：

$$p(c_i) = \frac{\text{类型为}\boldsymbol{C_i}\text{的文档个数}}{\text{训练集中文档总数}N}$$

为了估计 p (d | ci)，需要一个假设：Term 独立性假设。文档中每个 term 的出现是彼此独立的，基于这个假设，似然概率 p (d | ci) 的估计方法如下：假设文档 d 包含 nd 个 term：t1, t2, …, tnd 根据 Term 的独立性假设，有

$$p(d \mid c_i) = p(t_1, t_2, ..., t_{n_d} \mid c_i) = p(t_1 \mid c)\, p(t_2 \mid c) ... p(t_{n_d} \mid c) = \prod_{1 \le k \le n_d} p(t_k \mid c_i)$$

$$p(t_k \mid c_i) = \frac{t_k\text{在类型为}\boldsymbol{C_i}\text{的文档中出现的次数}}{\text{在类型为}\boldsymbol{C_i}\text{的文档中出现的term的总数}}$$

基于以上理论，实现朴素贝叶斯分类器即分为两步：第一步，通过训练集计算每个分类的先验概率、及每个词条属于不同分类的条件概率；第二步，测试集统计测试文档的词频，并利

用前面计算得到的先验概率和条件概率计算出每一个测试文档属于每个分类的概率，概率最大的分类即为预测的分类类型。

2. Map/Reduce 编程模型简介

Map/Reduce 是 Google 提出的一个软件架构，用于大规模数据集的并行运算。概念“Map（映射）”和“Reduce（归纳）”，及他们的主要思想，都是从函数式编程语言借来的，还有从矢量编程语言借来的特性。当前的软件实现是指定一个 Map（映射）函数，用来把一组键值对映射成一组新的键值对，指定并发的 Reduce（归纳）函数，用来保证所有映射的键值对中的每一个共享相同的键组。

简单来说，一个映射函数就是对一些独立元素组成的概念上的列表（例如，一个测试成绩的列表）的每一个元素进行指定的操作（比如，有人发现所有学生的成绩都被高估了一分，他可以定义一个“减一”的映射函数，用来修正这个错误。）。事实上，每个元素都是被独立操作的，而原始列表没有被更改，因为这里创建了一个新的列表来保存新的答案。这就是说，Map 操作是可以高度并行的，这对高性能要求的应用以及并行计算领域的需求非常有用。

而归纳操作指的是对一个列表的元素进行适当的合并（继续看前面的例子，如果有人想知道班级的平均分该怎么做？他可以定义一个归纳函数，通过让列表中的奇数（odd）或偶数（even）元素跟自己的相邻的元素相加的方式把列表减半，如此递归运算直到列表只剩下一个元素，然后用这个元素除以人数，就得到了平均分）。虽然他不如映射函数那么并行，但是因为归纳总是有一个简单的答案，大规模的运算相对独立，所以归纳函数在高度并行环境下也很有用。

Map/Reduce 通过把对数据集的大规模操作分发给网络上的每个节点实现可靠性；每个节点会周期性的把完成的工作和状态的更新报告回来。如果一个节点保持沉默超过一个预设的时间间隔，主节点记录下这个节点状态为死亡，并把分配给这个节点的数据发到别的节点。每个操作使用命名文件的不可分割操作以确保不会发生并行线程间的冲突；当文件被改名的时候，系统可能会把他们复制到任务名以外的另一个名字上去。

归纳操作工作方式很类似，但是由于归纳操作在并行能力较差，主节点会尽量把归纳操作调度在一个节点上，或者离需要操作的数据尽可能近的节点上了，因为他们有足够的带宽，他们的内部网络没有那么多的机器。

3. Map/Reduce 算法设计与实现

Map/Reduce 是 Google 提出的一个软件架构，用于大规模数据集的并行运算。概念“Map（映射）”和“Reduce（归纳）”，及他们的主要思想，都是从函数式编程语言借来的，还有从矢量编程语言借来的特性。当前的软件实现是指定一个 Map（映射）函数，用来把一组键值对映射成一组新的键值对，指定并发的 Reduce（归纳）函数，用来保证所有映射的键值对中的每一个共享相同的键组。

3.1 训练过程

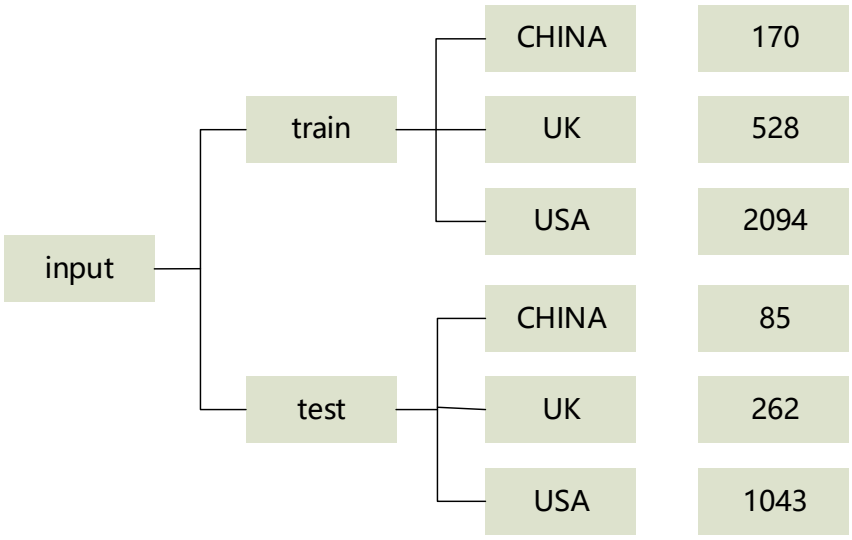
训练分为两个阶段，第一阶段统计每个分类的文档数以及所有文档总数，并计算每个分类的先验概率；第二阶段统计每个文档每个单词的词频，并统计词条总数，计算每个词条属于不同分类的条件概率。

由于计算先验概率，并不需要读取文档的内容，而只需要统计数量，所以在此没有使用 Map/Reduce 完成，直接采用 HDFS 提供的 FileSystem API 来完成元数据的提取（目录信息：目录名，包含文件个数等）。实现方法如下：

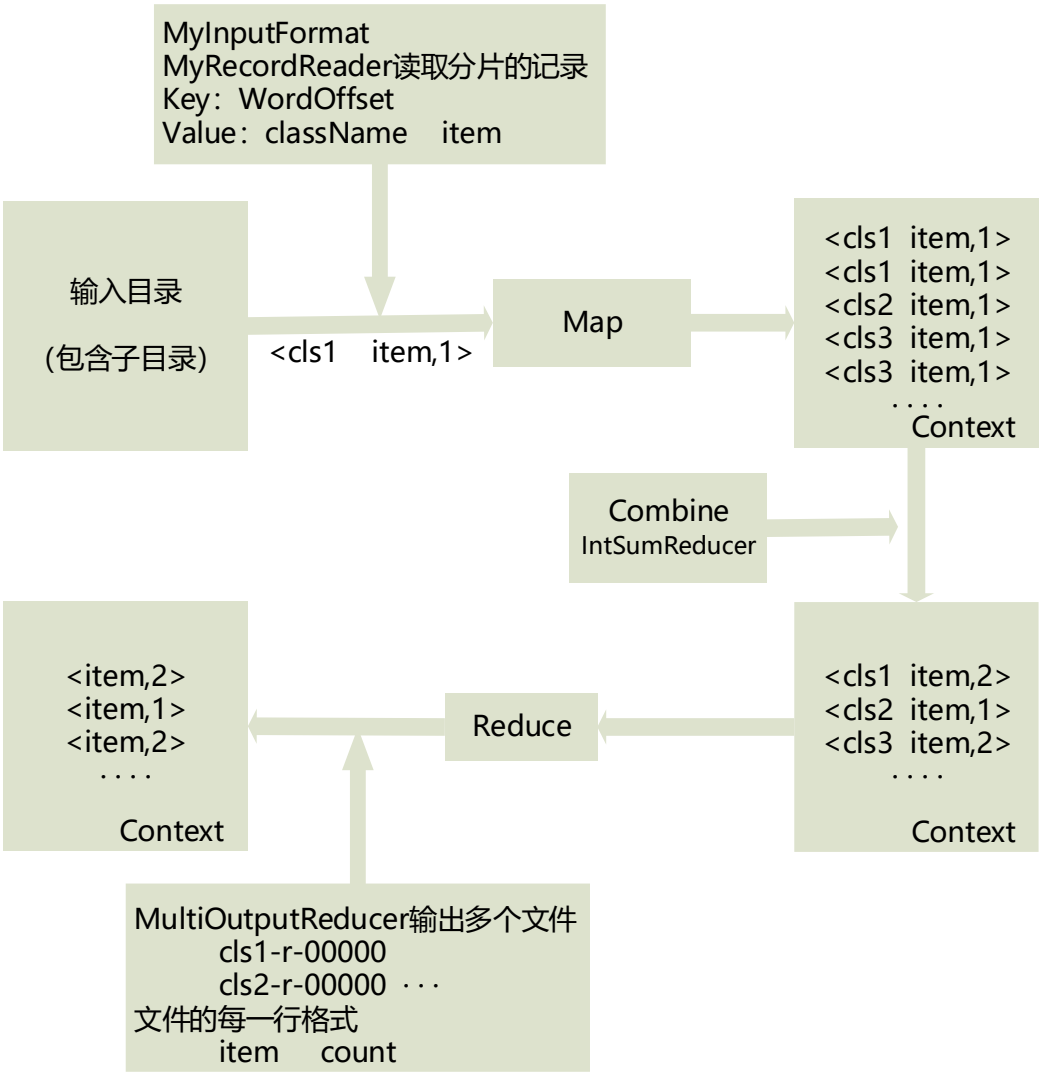
遍历训练数据集目录，得到每个分类的文档数。同时，将分类和文档数保存到 HashMap 中，并累计得到文档总数。最后直接遍历 HashMap，取得文档数除以总数即得先验概率，以<类别，概率>键值对格式存储，保存到 prior.txt 结果文件中（概率保留八位小数）。

计算完先验概率后将结果保存在内存中，再启动 MR 任务计算每个词条的条件概率，需要先计算词频、分类的词条总数以及字典大小，然后利用上述格式计算，并注意词频为零的情况的处理。

MR 程序的输入为一个结构化的目录，目录层次如下图所示：



MR 的流程图如下所示：



输入的文档分散在不同目录，需要进行递归处理。

由于都是小文件，直接读取效率低下，所以考虑用 CombineFileInputFormat 来批量读取文件，并且需要实现一个 RecordReader 用来读取文件的每一条记录。直接 daoruorg.apache.hadoop.examples.MultiFileWordCount，该类主要实现了三个子类，WordOffset 子类实现了 WritableComparable 接口，保留了文件名以及在分片中的偏移值；MyInputFormat 子类用来创建记录阅读器；最核心的是 CombineFileLineRecordReader 子类，实现了如何从 CombineFileSplit 中读取一条记录，并返回 KeyValue 对。所以，直接继承 CombineFileLineRecordReader 子类，并实现自己的新功能：读取文档所属分类目录。

经过 map 输出后格式为<className item, count>，reduce 的主要任务就是完成统计功能，实现如下：其中：dictSet 类型为 HashSet<item>用于存储字典集合，classSize 类型为

HashMap<className, itemCount>用于存储分类词条总数，classItemCount 类型为 HashMap<className, HashMap<item, count>>用于存储每个分类的每个词条的词频。

若 MR 正确完成，则所需数据均已保存到内存中，有 docsMap, priorMap, dictSet, classSize, classItem。接下来只需遍历他们，计算词条的条件概率，并将结果保存到文件即可。condMap 类型为 HashMap<item, class1:p1 class2:p2 …>用于保存词条分别属于分类的概率。

遍历上述 HashMap 将结果写入文件，至此，训练结束。

3.2 测试过程

测试过程与训练过程处理逻辑大同小异，都是处理结构化的目录及文档，结构图在训练过程展示。不同的是，训练提取目录作为分类名称，而测试的目录则为实际分类，并且还需要提取文件的名称来标识不同文件。流程图与上述类似不再赘述，下面给出不同之处的实现。

MyRecordReader 需要完成两个功能，一个是读取文档真实类型，另一个是读取文档路径即名称。经过 map 输出后格式为<className-pathname, item>，reduce 的主要任务就是完成后验概率的计算。priorMap 为预先读取的分类先验概率，sizeMap 为训练结果的 classSize 保存分类词条总数以及字典大小，dictSet 为字典集合，resultMap 类型为 HashMap<className-docName, HashMap<classA, pA>用于保存每个文档属于不同分类的概率值。

MR 测试完成后，只需遍历 resultMap 即可得出每个文档的真实类型和预测分类，写入文件保存结果即可。

预测结果文件的行记录格式为：

<docName realClass predClass class1:p1 class2:p2 …>

4. 实验结果分析

```
USA
TP:1021     FP:29
FN:22       TN:318
P:0.9724    R:0.9789    F1:0.9756

CHINA
TP:69       FP:9
```

```
FN:16        TN:1296
P:0.8846     R:0.8118     F1:0.8466

UK
TP:237       FP:25
FN:25        TN:1103
P:0.9046     R:0.9046     F1:0.9046

Micro Avg
P:0.9205     R:0.8984     F1:0.9089

Macro Avg
TP:1327      FP:63
FN:63        TN:2717
P:0.9547     R:0.9547     F1:0.9547
```

5. 总结与改进

1、起初设计思路是，MR 仅仅用来统计词频，所有的计算都是统计完成后读取 MR 输出文件结果，再进行计算，耗费了大量的 IO 时间，而且没有充分利用 reduce 的功能。改进后的实现如前所述，将所有中间结果保存在内存中，在 reduce 函数内进行循环迭代，不断更新。

2、起初过分依赖于两个确切的分类，如 UK、USA，改进后，将分类信息保存到 HashMap，每次从中读取，因此对于分类的数量没有限制。三个、四个类别均可通用，实现结果基于三个类别 CHINA、UK、USA。

3、本实验唯一的约束条件在于输入数据的组织结构，即前述的目录结构，文档都归类到某一个目录下，否则，需要修改 MR 实现方式。本文的实现方式使用了训练文档的路径信息，根据路径判断训练文档的类别，以及记录测试文档的真实类别，用于计算预测精度。实际上测试文档可不必按此归类，那就需要通过其他方式获取真实类别。

参考文献

[1]王国才. 朴素贝叶斯分类器的研究与应用[D].重庆交通大学,2010.

[2]蒋良孝. 朴素贝叶斯分类器及其改进算法研究[D].中国地质大学,2009.

[3]李静梅,孙丽华,张巧荣,张春生. 一种文本处理中的朴素贝叶斯分类器[J]. 哈尔滨工程大学学报,2003,(01):71-74.

[4]维基百科