# MapReduce数据挖掘基础算法(II)

分类算法并行化


# 摘要



- 分类问题的基本描述
- KNN最邻近分类算法
- 朴素贝叶斯分类算法
- 决策树分类算法
- 支持向量机分类算法


# 聚类

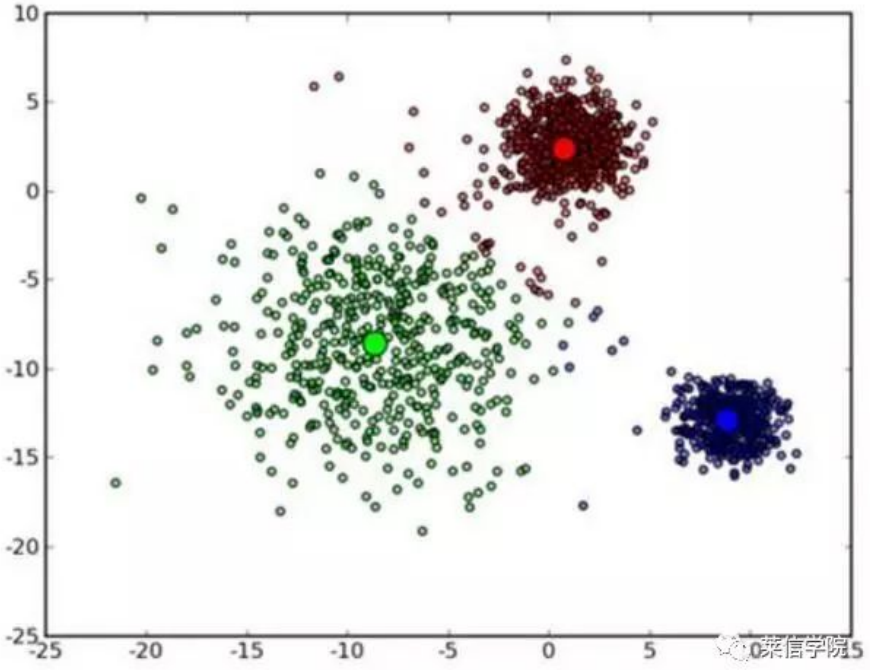


- 聚类与分类的不同在于，聚类所要求划分的类是*未知*的。聚类的目的是使得属于同类别的对象之间的差别尽可能的小，而不同类别上的对象的差别尽可能的大。因此，聚类的意义就在于将观察到的内容依据相应算法组织成类分层结构，把类似的事物组织在一起。
- 在机器学习中，聚类是一种无监督学习。也就是说，聚类是在预先不知道欲划分类的情况下，根据信息相似度原则进行信息聚类的一种方法。通过聚类，人们能够识别密集的和稀疏的区域，因而发现全局的分布模式，以及数据属性之间的有趣的关系。

# 聚类



- 从统计学的观点看，聚类分析是通过数据建模简化数据的一种方法。常见的聚类算法包括：K-Means聚类算法、K-中心点聚类算法、层次聚类、DBScan、EM聚类等。




# 分类

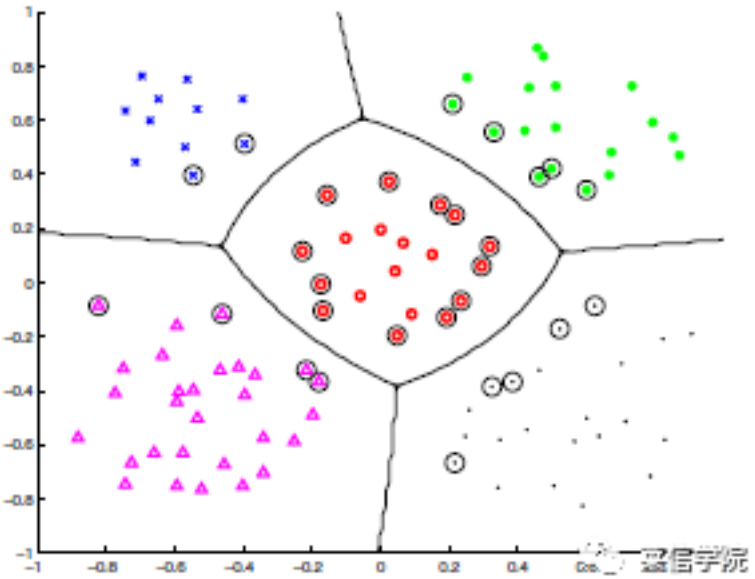


- 分类（Classification）是机器学习和数据挖掘中最重要、最频繁使用的算法。
- 分类算法的基本作用是：从一组已经带有分类标记的训练样本数据集来预测一个测试样本的分类结果。
- 从机器学习的观点，分类属于监督学习，每个训练样本的数据对象已经有类标识，通过学习可以形成表达数据对象与类标识间对应的知识。
- 分类具有广泛的应用，例如医疗诊断、信用卡的信用分级、图像模式识别、营销用户画像等。

# 分类



- 分类挖掘所获的分类模型可以采用多种形式加以描述输出。其中主要的表示方法有：分类规则、决策树、数学公式和神经网络等。




# 分类 vs. 聚类



- 分类
  - 监督学习
  - 类别已知，通过对已知分类的数据进行训练和学习，找到不同类的特征，再对未分类的数据进行分类
  - 根据文本的特征或属性，划分到已有类别中。
- 聚类
  - 无监督学习
  - 类别未知，通过聚类分析将数据聚合成几个群体
  - 需要分析人员找出各类用户的重要特征
  - 需要通过各类别的特征解释含义以及为各类别命名。

# 分类问题基本描述



- 分类算法的基本描述：
  - 一个训练数据集TR={tr1, tr2, tr3, …}
  - 每个训练样本tri是一个三元组(tid, A, y)
  - 其中tid是每个训练样本的标识符，A是一组特征属性值：A={a1, a2, a3, …}，而y是训练样本已知的分类标记。

| tid | age | sex | cm | kg | 发育 |
|---|---|---|---|---|---|
| 1 | 2 | M | 80 | 14 | 良好 |
| 2 | 3 | M | 82 | 12 | 不良 |
| 3 | 2 | F | 68 | 12 | 良好 |
| … | … | … | … | … | … |


# 分类问题基本描述



- 对于一个测试样本数据集TS={ts1, ts2, ts3, ...}，每个测试样本ts也是一个三元组(tid, A, y')，其中y'未知。
- 需要解决的问题是：根据训练数据集来预测出每个ts的未知的分类标记y'。
- 训练数据集越大，对测试样本分类标记的预测越准确。

# 分类问题基本描述



训练样本数据

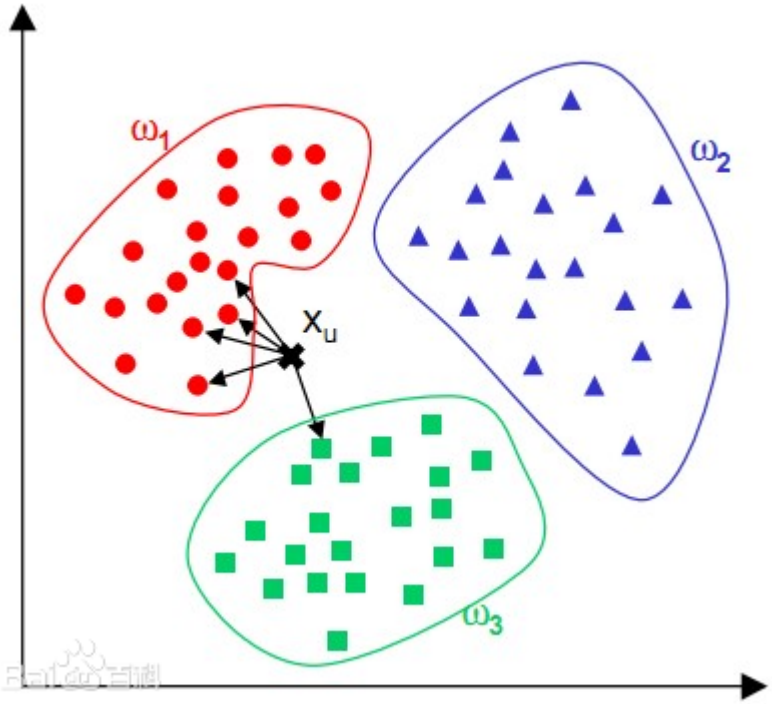
| tid | age | sex | cm | kg | 发育 |
|---|---|---|---|---|---|
| 1 | 2 | M | 80 | 14 | 良好 |
| 2 | 3 | M | 82 | 12 | 不良 |
| 3 | 2 | F | 68 | 12 | 良好 |
| 4 | 2 | F | 75 | 17 | 肥胖 |
| … | … | … | … | … | … |

测试样本数据

| tid | age | sex | cm | kg | 发育 |
|---|---|---|---|---|---|
| 1 | 2 | F | 70 | 15 | ? |
| 2 | 2 | M | 82 | 12 | ? |
| 3 | 3 | F | 68 | 12 | ? |
| 4 | 2 | M | 75 | 17 | ? |
| … | … | … | … | … | … |

# KNN算法






# K-最邻近(KNN)分类并行化算法



- 基本算法设计思想
  - K-最近邻是分类器算法中最通俗易懂的一种，计算测试样本到各训练样本的距离，取其中距离最小的K个，并根据这K个训练样本的标记进行投票得到测试样本的标记。
  - 加权K-最近邻分类算法的思路是，在根据测试样本的标记进行投票表决时，将根据测试样本与每个训练样本间距离（或相似度）的大小决定训练样本标记的作用大小，基本原则是：距离越近的训练样本其标记的作用权重越大，反之则越小。据此，可以建立一个带加权的投票表决计算模型(比如y' = ΣSi*yi/ΣSi, k=[0,k-1],Si为取值0-1的相似度数值，yi为选取出的最邻近训练样本的分类标记值)决定以最终的测试样本的分类标记。
- 算法的思路清晰简单，然而对于海量数据计算量很大，耗费时间较长。


# MapReduce并行化算法设计思路



- 基本处理思路是：将测试样本数据分块后分布在不同的节点上进行处理，将训练样本数据文件放在DistributedCache中供每个节点共享访问。
- Map阶段对每个读出的测试样本数据ts(trid, A', y')
  - 计算其与每个训练样本数据tr(trid, A, y)之间的相似度S=Sim(A', A）（1：相似度最大，0：相似度最小）
  - 检查S是否比目前的k个S值中最小的大，若是则将(S, y)计入k个最大者
  - 根据所保留的k个S值最大的(S, y)，根据模型y' =ΣSi*yi/Σsi计算出ts的分类标记值y'，发射出(tsid, y')
- Reduce阶段直接输出(tsid, y')

# MapReduce并行化算法实现

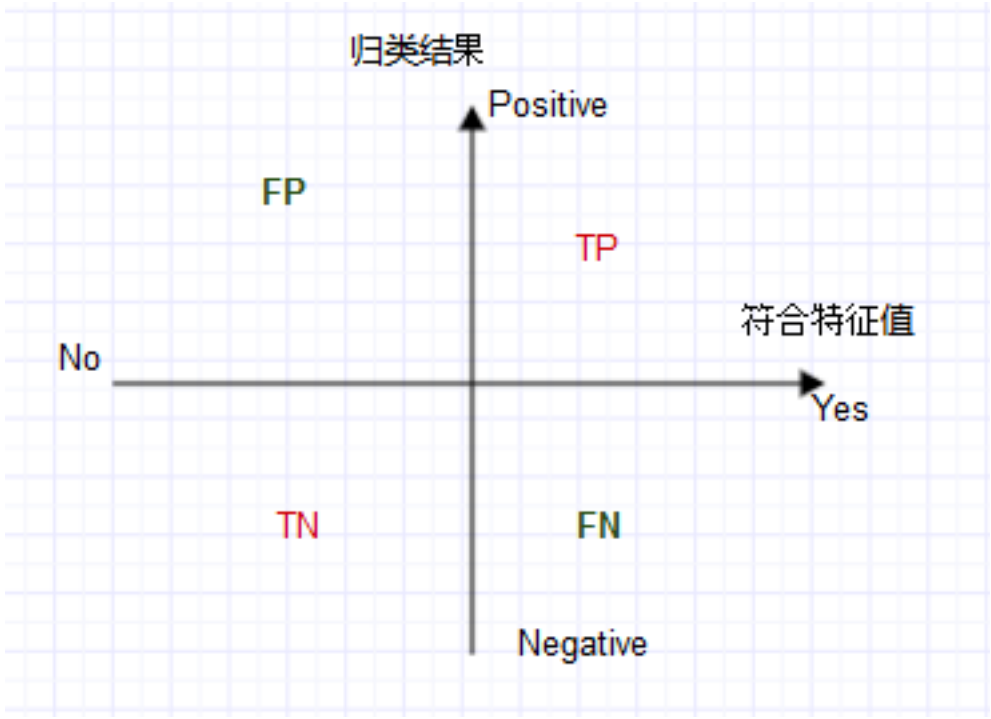


- Mapper伪代码

```
class Mapper{
    setup(…){ //读取全局训练样本数据文件，转入本地内存的数据表TR中 }
    map(key, ts) {// ts为一个测试样本
      Φ → MaxS(k)
      ts →tsid, A', y'
      for i=0 to TR.lenghth {
          TR[i] → trid, A, y
          S = Sim(A, A');
          若S属于k个最大者，(S, y) → MaxS;
      }
      根据MaxS和带加权投票表决模型计算出y' =ΣSi*yi/ΣSi
      emit(tsid, y')
  }
}
```

# 分类算法的评估





# 分类算法的评估



- Accuracy准确率
  - 对于给定的测试数据集，分类器正确分类的样本数与总样本数之比

$$A(M) = \frac{TN + TP}{TN + FP + FN + TP}$$

- Precision精确率
  - 预测结果中符合实际值的比例，可以理解为没有“误报”的情形

$$P(M) = \frac{TP}{TP + FP}$$

# 分类算法的评估



- Recall召回率
  - 正确分类的数量与所有“应该”被正确分类（符合目标标签）的数量的比例，可以理解为召回率对应的没有“漏报”的情形。

$$R(M) = \frac{TP}{TP + FN}$$

- F1 Score：精确率和召回率的调和均值

$$\frac{2}{F1} = \frac{1}{P} + \frac{1}{R} \quad \Rightarrow \quad F1 = \frac{2TP}{2TP + FP + FN}$$

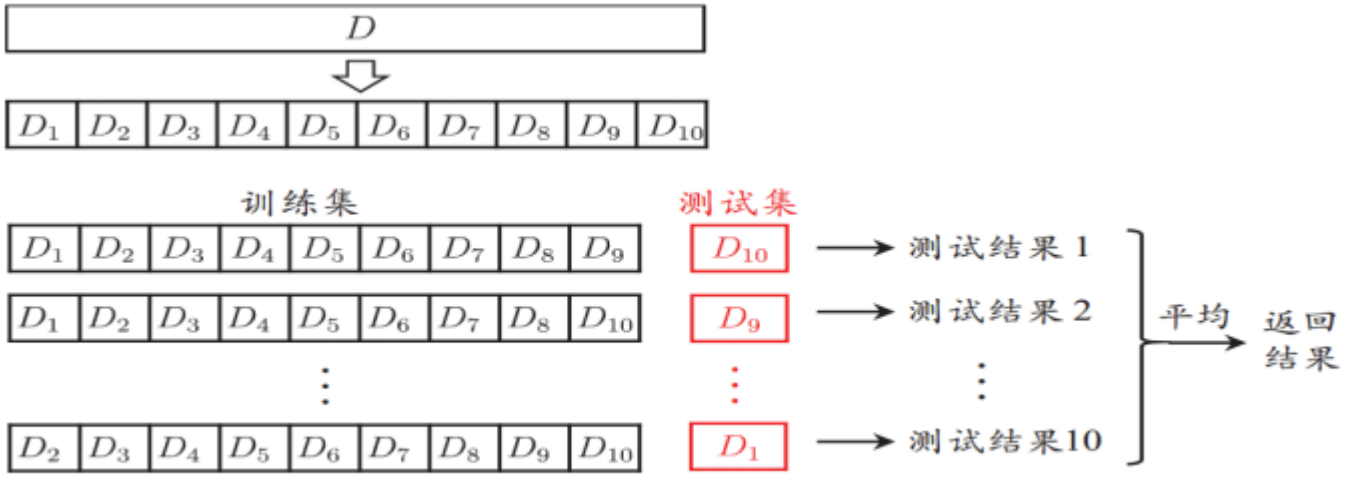

# 分类算法的评估方法



- 通常将包含m个样本的数据集$D=\{(x_1,y_1),(x_2,y_2),\ldots,(x_m,y_m)\}$拆分成训练集S和测试集T：
- 留出法：
  - 直接将数据集划分为两个互斥集合
  - 训练/测试集划分要尽可能保持数据分布的一致性
  - 一般若干次随机划分、重复实验取平均值
  - 训练/测试样本比例通常为2:1~4:1
- 交叉验证法：
  - 将数据集分层采样划分为k个大小相似的互斥子集，每次用k-1个子集的并集作为训练集，余下的子集作为测试集，最终返回k个测试结果的均值，k最常用的取值是10。

# 分类算法的评估方法






10 折交叉验证示意图

- 为了减小因样本划分不同而引入的差别，k折交叉验证通常随机使用不同的划分重复p次，最终的评估结果是这p次k折交叉验证结果的均值，例如常见的“10次10折交叉验证”。
- 假设数据集D包含m个样本，若令k=m，则得到留一法。不受随机样本划分方式的影响，结果往往比较准确，但当数据集比较大时，计算开销难以忍受。


# 贝叶斯定理



- 在一个论域中，某事件A发生的概率用P(A)表示，事件的条件概率P(A|B)的定义为：在事件B已经发生的前提下事件A发生的概率。
  - P(A|B) = P(A)*P(B|A)/P(B)
- 朴素贝叶斯分类器基于一个简单的假定：给定目标值时属性之间相互条件独立。

# 朴素贝叶斯分类算法



- 基本问题描述
  - 设每个数据样本用一个$n$维特征向量来描述$n$个属性的值，即：$X=\{x1,x2,\dots,xn\}$，假定有$m$个类，分别用$Y1,Y2,\dots Ym$表示
  - 给定一个未分类的数据样本$X$，若朴素贝叶斯分类将未知的样本$X$分配给类$Yi$，则一定有$P(Yi|X)>P(Yj|X)$，$1\le j\le m,\ j\ne i$
  - 根据贝叶斯定理
  - $P(Yi|X)=\frac{P(Yi)*P(X|Yi)}{\sum_{i=1}^{k}P(Yi)*P(X|Yi)}=\frac{P(Yi)*P(X|Yi)}{P(X)}$
  - 由于$P(X)$对于所有类为常数，概率$P(Yi|X)$可转化为概率$P(X|Yi)*P(Yi)$。


# 朴素贝叶斯分类算法



- 如果训练数据集中有很多具有相关性的属性，计算$P(X|Yi)$将非常复杂，为此，通常假设各属性是互相独立的，这样$P(X|Yi)$的计算可简化为求$P(x1|Yi)$，$P(x2|Yi)$，…，$P(xn|Yi)$之积；而每个$P(xj|Yi)$可以从训练数据集近似求得。

- 据此，对一个未知类别的样本$X$，可以先分别计算出$X$属于每一个类别$Yi$的概率$P(X|Yi)*P(Yi)$，然后选择其中概率最大的$Yi$作为其类别。

# 朴素贝叶斯分类并行算法



- 哪些部分可以并行，而结果又如何汇总？
- 根据前述的思路，判断一个未标记的测试样本属于哪个类$Yi$的核心任务成为：根据训练数据集计算$Yi$出现的频度和所有属性值$xj$在$Yi$中出现的频度。
- 据此，并行化算法设计的基本思路是：用MapReduce扫描训练数据集，计算每个分类$Yi$出现的频度$FYi$(即$P(Yi)$)、以及每个属性值出现在$Yi$中的频度$FxYij$ (即$P(xj|Yi)$)

# 朴素贝叶斯分类并行算法



- 而在MapReduce中对训练数据集进行以上的频度计算时，实际上就是简单地统计$Yi$和每个$xj$在$Yi$中出现的频度
- 在进行分类预测时，对一个未标记的测试样本$X$，根据其包含的每个具体属性值$xj$，根据从训练数据集计算出的$FxYij$进行求积得到$FxYij$(即$P(X|Yi)$，再乘以$FYi$即可得到$X$在各个$Yi$中出现的频度$P(X|Yi) * P(Yi)$，取得最大频度的$Yi$即为$X$所属的分类。此过程在Map阶段实现。
- Reduce过程只需将各节点的统计频度汇总。

# 朴素贝叶斯分类并行算法



- 训练算法的Mapper伪代码

```
//输入数据：整个训练样本数据集
//输出数据：各Map节点输出的局部频度数据FYi和FxYij
class TrainMapper{
   map(key, TR){
      //TR为一个训练样本，由一个样本标志trid、属性x以及分类值y组成
      TR→trid, X, y
      emit(y,1)
      for j=0 to X.length(){
         X[j] → 属性名xnj和属性值xvj
         emit(<y, xnj, xvj>, 1)
      }
   }
}
```

# 朴素贝叶斯分类并行算法



- 训练算法的Reducer伪代码

```
//输入数据：Map节点输出的局部频度数据FYi和FxYij
//输出数据：全局的频度数据FYi和FxYij
class TrainReducer{
   reduce(key, value_list){
      //key或为分类标记y，或为<y, xnj, xvj>
      //value_list中的每个值是key出现的次数
      //累加后即为Fyi或者FxYij
      //reduce只是简单地汇总两个频度即可
       sum = 0
       while (value_list.hasNext()){  sum+= value_list.next().get();  }
       emit(key, sum)
   }
}    //Reduce完成后，FYi和FxYij频度数据将输出并保存到HDFS文件中
```

# 朴素贝叶斯分类并行算法



- 分类预测算法的Mapper伪代码

```
//输入数据：测试样本数据集
//输出数据：各测试样本及其分类结果
class TestMapper{
    setup(…){
        //初始化时读取从训练数据集得到的频度数据
        //分别装入频度表FY和FxY供各个map节点共享访问
        //分类频度表FY = { (Yi, 每个Yi的频度FYi) }
        //属性频度表FxY= { (<Yi, xnj, xvj>, 出现频度FxYij)}
    }
```

# 朴素贝叶斯分类并行算法



```
map(key, ts){ // ts为一个测试样本
   ts →tsid, A
   MaxF= MIN_VALUE; idx= -1;
   for (i=0 to FY.length){
      FXYi= 1.0; Yi = FY[i].Yi; FYi= FY[i].FYi;
      for (j=0 to A.length){
         xnj= A[j].xnj; xvj= A[j].xvj
         根据<Yi, xnj, xvj>扫描FxY表, 取得FxYij
         FXYi= FXYi* FxYij;
      }//执行到此，FXYi等价于公式中的P(X|Yi)
      if(FXYi* FYi>MaxF) { MaxF= FXYi*FYi; idx= i; }
   }
  emit(tsid, FY[idx].Yi)
}
}
```


# 决策树

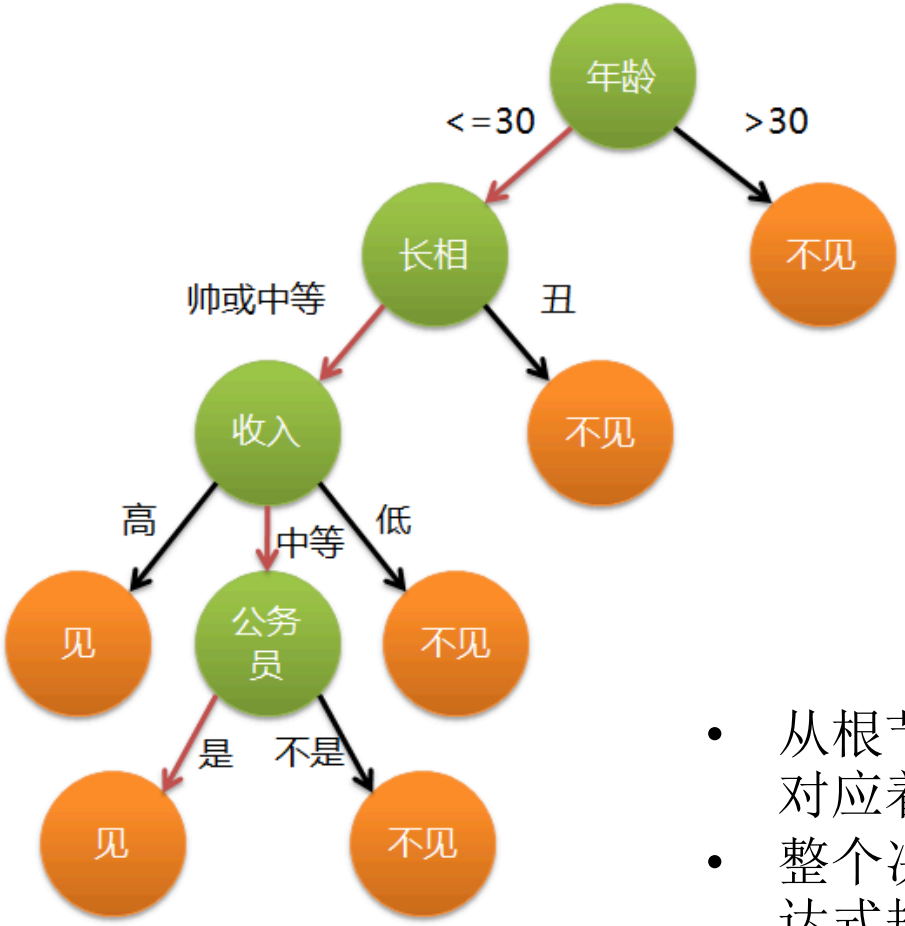


- 决策树基于“树”结构进行决策。
  - 每个“内部节点”对应于某个属性上的“测试”
  - 每个分支对应于该测试的一种可能结构（即该属性的某个取值）
  - 每个“叶结点”对应于一个“预测结果”
- 决策树是以实例为基础的归纳学习算法。
- 学习过程：通过对训练样本的分析来确定“划分属性”（即内部节点对应的属性）。
- 预测过程：将测试示例从根节点开始，沿着划分属性所构成的“判定测试序列”下行，直到叶结点。

引自： 机器学习，周志华，清华大学出版社，2016年1月

# 决策树






- 从根节点到叶节点的一条路径对应着一条合取规则；
- 整个决策树对应着一组析取表达式规则。


# 决策树算法



- CLS(Concept Learning System)，1966年
- ID3，1979年
- C4.5，1993年
- CART(Classification and Regression Tree)，1984年
- RF(Random Forest)，2001年


# 基本流程



- 策略“分而治之”
- 自根至叶的递归过程
- 在每个中间节点寻找一个“划分”(split or test)属性
- 三种停止条件：
  - 当前结点包含的样本全属于同一类别，无需划分；
  - 当前属性集为空，或是所有样本在所有属性上取值相同，无法划分；
  - 当前结点包含的样本集合为空，不能划分。

输入：训练集D，属性集A

过程：函数TreeGenerate(D, A)

1: 生成结点node；

2: if D中样本全属于同一类别C then

3: 将node标记为C类叶结点； return

4: end if

5: if A = ∅ or D中样本在A上取值相同 then

6: 将node标记为叶结点，其类别标记为D中样本数最多的类； return

7: end if

8: 从A中选择最优划分属性$a_*$；

9: for $a_*$的每个值$a_*^v$ do

10: 为node生成一个分支；另$D_v$表示D中在$a_*$上取值为$a_*^v$的样本子集；

11: if $D_v$为空 then

12: 将分支结点标记为叶结点，其类别标记为D中样本最多的类； return

13: else

14: 以TreeGenerate($D_v$, A\{$a_*$}) 为分支结点

15: end if

16: end for

输出：以node为根节点的一棵决策树

利用当前结点的后验分布

决策树算法的核心

将父结点的样本分布作为当前结点的先验分布


# 信息增益（information gain）



- 信息熵（entropy）是度量样本集合“纯度”最常用的一种指标。
- 假定当前样本集合$D$中第$k$类样本所占的比例为$p_k$，则$D$的信息熵定义为

$$Ent(D) = -\sum_{k=1}^{|y|} p_k \log_2 p_k$$

$Ent(D)$的值越小，则$D$的纯度越高

信息增益直接以信息熵为基础，计算当前划分对信息熵所造成的变化

# 信息增益



- 离散属性a的取值：$\{a^1, a^2, \ldots, a^v\}$
- $D^v$：D中在a上取值=$a^v$的样本集合
- 以属性a对数据集D进行划分所获得的信息增益为：

$$Gain(D,a) = Ent(D) - \sum_{v=1}^{V} \frac{|D^v|}{|D|} Ent(D^v)$$

划分前的信息熵

划分后的信息熵

第v个分支的权重，
样本越多越重要

ID3算法中使用

# 增益率（Gain Ratio）

- 信息增益：对可取值数目较多的属性有所偏好
  - 有明显弱点，例如：将“编号”作为一个属性
- 增益率：

$$\text{Gain\_}Ratio(D,a) = \frac{Gain(D,a)}{IV(a)}$$

$$\text{其中IV}(a) = -\sum_{v=1}^{V} \frac{|D^v|}{|D|} \log_2 \frac{|D^v|}{|D|}$$

属性a的可能取值数目越多（即V越大），则IV(a)的值通常就越大。

- 启发式：先从候选划分属性中找出信息增益率高于平均水平的，再从中选取增益率最高的。

C4.5算法中使用

# 基尼指数（gini index）



$$Gini(D) = \sum_{k=1}^{|y|} \sum_{k' \neq k} p_k \, p_{k'} = 1 - \sum_{k=1}^{|y|} p_k^2$$

反映了从$D$中随机抽取两个样例，其类别标记不一致的概率

$Gini(D)$越小，数据集$D$的纯度越高。

- 属性$a$的基尼指数：

$$\text{Gini\_}index(D, a) = \sum_{v=1}^{V} \frac{|D^v|}{|D|} Gini(D^v)$$

在候选属性集合中，选取那个使划分后基尼指数最小的属性

# Example

- Class P: buys_computer = "yes"
- Class N: buys_computer = "no"

$$Info(D) = I(9,5) = -\frac{9}{14}\log_2(\frac{9}{14}) - \frac{5}{14}\log_2(\frac{5}{14}) = 0.940$$

| age | $p_i$ | $n_i$ | $I(p_i, n_i)$ |
|---|---|---|---|
| <=30 | 2 | 3 | 0.971 |
| 31…40 | 4 | 0 | 0 |
| >40 | 3 | 2 | 0.971 |

| age | income | student | credit_rating | buys_computer |
|---|---|---|---|---|
| <=30 | high | no | fair | no |
| <=30 | high | no | excellent | no |
| 31…40 | high | no | fair | yes |
| >40 | medium | no | fair | yes |
| >40 | low | yes | fair | yes |
| >40 | low | yes | excellent | no |
| 31…40 | low | yes | excellent | yes |
| <=30 | medium | no | fair | no |
| <=30 | low | yes | fair | yes |
| >40 | medium | yes | fair | yes |
| <=30 | medium | yes | excellent | yes |
| 31…40 | medium | no | excellent | yes |
| 31…40 | high | yes | fair | yes |
| >40 | medium | no | excellent | no |

$$Info_{age}(D) = \frac{5}{14}I(2,3) + \frac{4}{14}I(4,0) + \frac{5}{14}I(3,2) = 0.694$$

$\frac{5}{14}I(2,3)$ means "age <=30" has 5 out of 14 samples, with 2 yes's and 3 no's. Hence

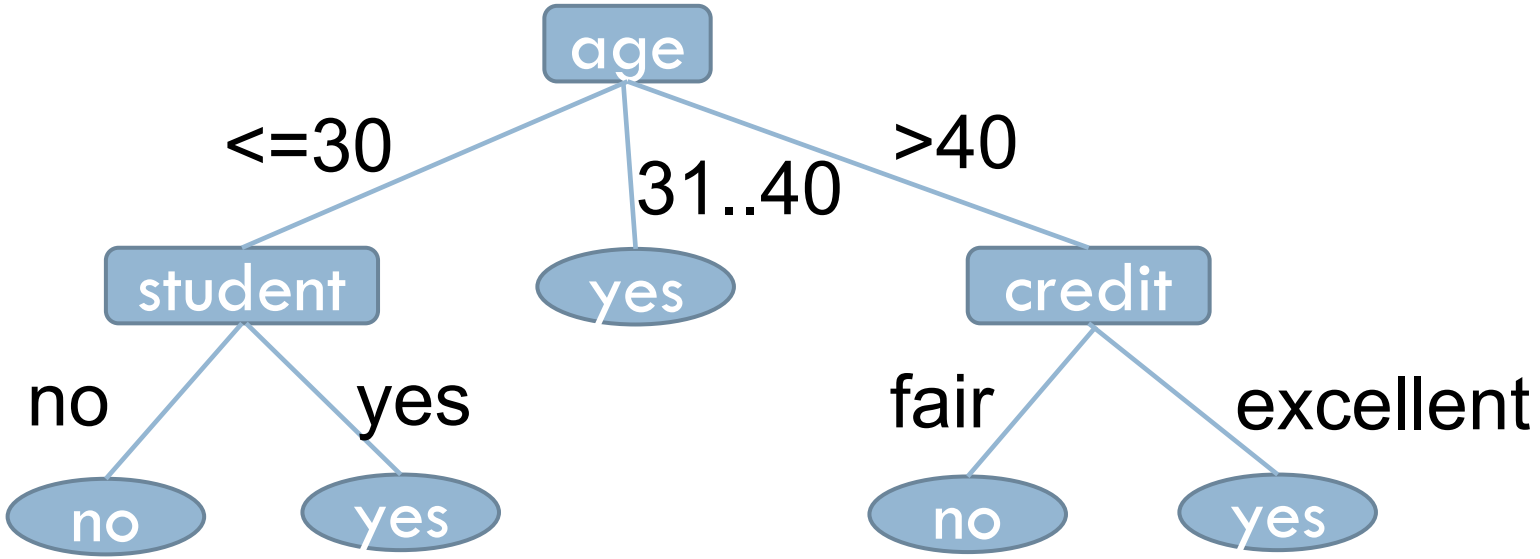
$$Gain(age) = Info(D) - Info_{age}(D) = 0.246$$

Similarly,

$$Gain(income) = 0.029$$
$$Gain(student) = 0.151$$
$$Gain(credit\_rating) = 0.048$$

# Example




age
<=30
31..40
>40
student
yes
credit
no
yes
fair
excellent
no
yes
no
yes


# 更多问题



- 过拟合
- 剪枝：预剪枝 vs. 后剪枝
- 连续值：连续值 vs. 缺失值
- 多变量决策树
- ……

本节内容只涉及原始决策树的生成算法


# 决策树并行化算法



- 基本设计思路
  - 关键任务：属性选择度量
  - 选取最佳“分裂”属性是整个决策树生成中占用计算机资源最大的阶段→利用MapReduce对这个阶段进行最大化的并行计算
  - 信息增益率计算是基于属性间相互独立的特点，可以利用MapReduce并行地统计计算增益率所需要的各个属性的相关信息。最后在构造决策树的主程序中利用这些统计好的信息快速地计算出属性的增益率，并选取最佳分裂属性。


# 决策树主程序设计



- 1. 执行构造决策树的串行算法（比如ID3）
  - 基于广度优先的生成策略
- 2. 在决策树构造算法需要计算信息增益率时，调用MapReduce过程在大规模的训练样本上进行统计，获得各个属性的统计信息，然后利用这些信息计算出属性的信息增益率。
  - 根据哈希表保存的统计信息，计算出每个节点的最佳分裂属性，然后对每个节点进行分裂。
  - 通过两个划分条件队列保存树节点信息。


# Map阶段设计



- 主要任务：对输入的大规模训练样本按照决策树中某一层节点的划分条件进行切分
  - 划分条件是该树节点在决策树中已经生成的路径
  - 决策树路径的构造方法基于层次切分数据的广度优先策略。
- Map函数
  - Key：标记不同树节点的临时nid、决策树表的某个属性S、该元组对应属性S的值s以及该元组的所属决策类c
  - Value：1

# Reduce阶段设计



- 完成对Map输出的<key, value>进行整理，将带有相同key值的value值累加得到value_sum。同时，将统计好的<key, value_sum>输出到分布式文件系统HDFS的文件中，以供主控程序计算各个属性的信息增益率的时候使用。


# 决策树并行化算法的实现



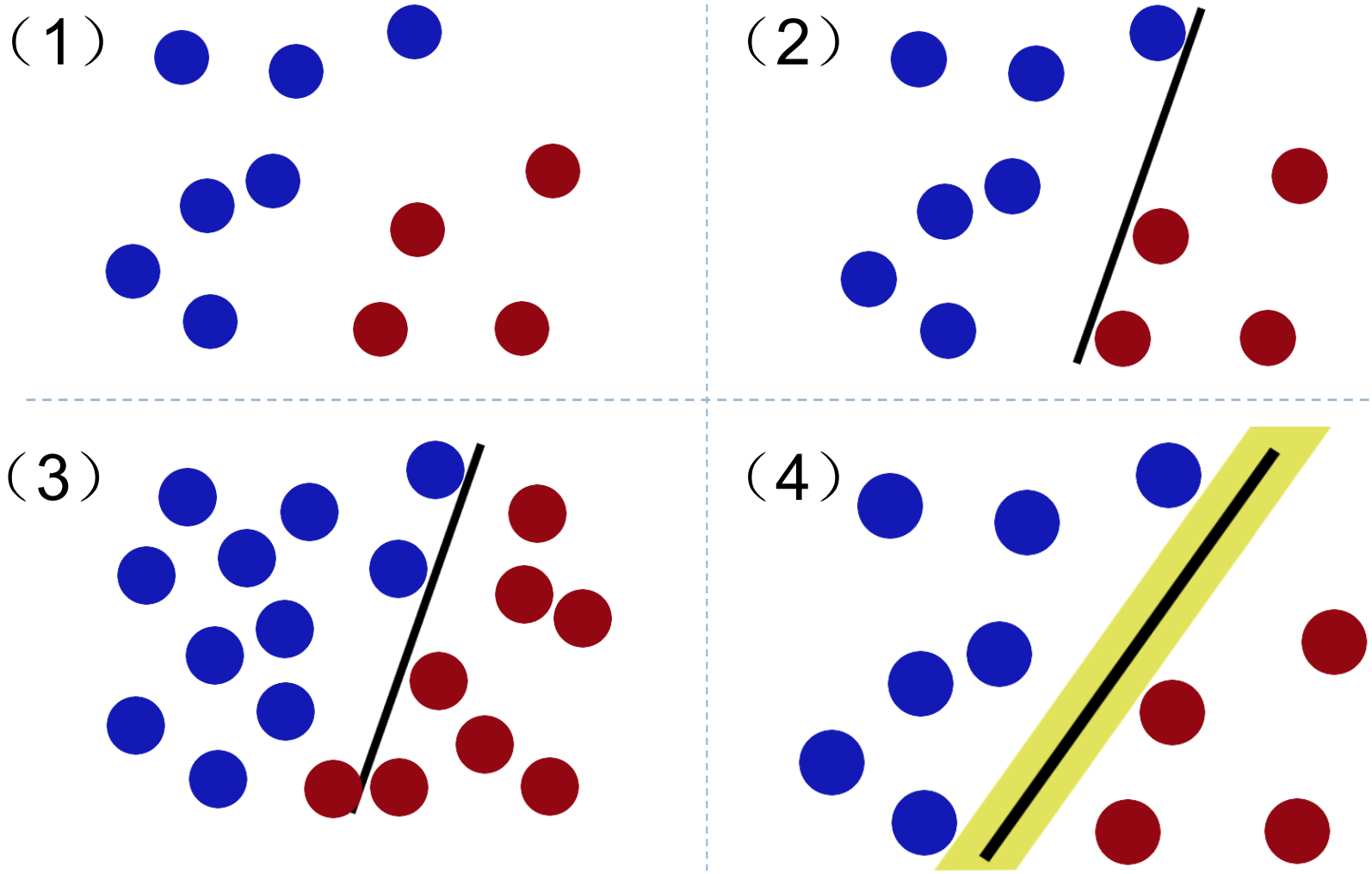
- 规则数据结构（即划分条件）
- 决策树主控程序DecisionTreeDriver
  - MapReduce作业配置和执行控制程序
  - 决策树算法主控程序
- DecisionTreeMapper
  - 对输入的训练样本按照划分条件进行切分的处理，中间结果发射到Reduce端。
- DecisionTreeReducer
  - 对Map端发射过来的各个属性下的零散信息，按照相同key值进行累加统计，并将最后统计的结果写入HDFS中，供主控程序计算信息增益率使用。


# 支持向量机



- SVM(Support Vector Machine)指的是支持向量机，是常见的一种判别方法。在机器学习领域，是一个有监督的学习模型，通常用来进行模式识别、分类以及回归分析。
- SVM的主要思想可以概括为两点：
  - 它是针对线性可分情况进行分析，对于线性不可分的情况，通过使用非线性映射算法将低维输入空间线性不可分的样本转化为高维特征空间使其线性可分，从而使得高维特征空间采用线性算法对样本的非线性特征进行线性分析成为可能。
  - 它基于结构风险最小化理论之上在特征空间中构建最优超平面，使得学习器得到全局最优化，并且在整个样本空间的期望以某个概率满足一定上界。

# SVM

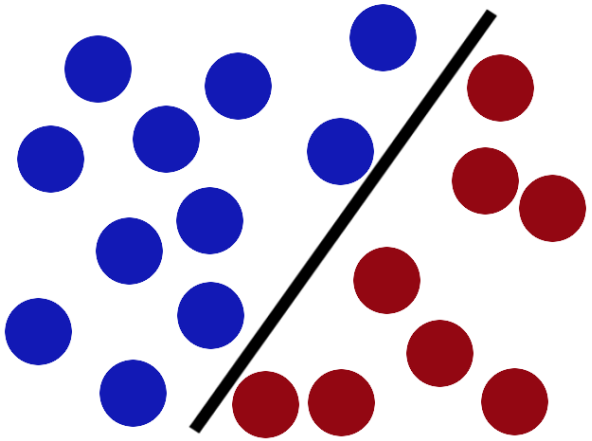



引自： https://www.reddit.com/r/MachineLearning/comments/15zrpp/please_explain_support_vector_machines_svm_like_i/

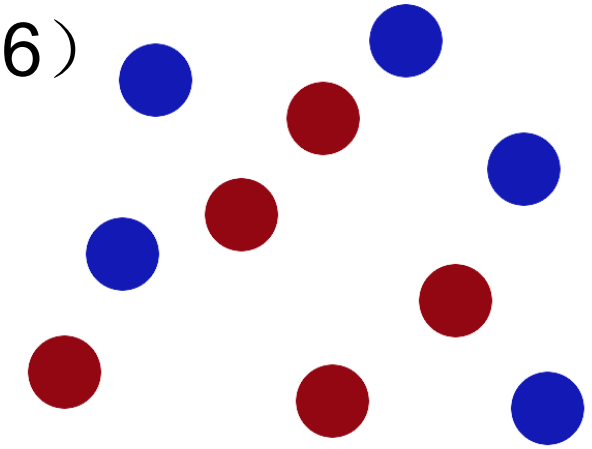
# SVM



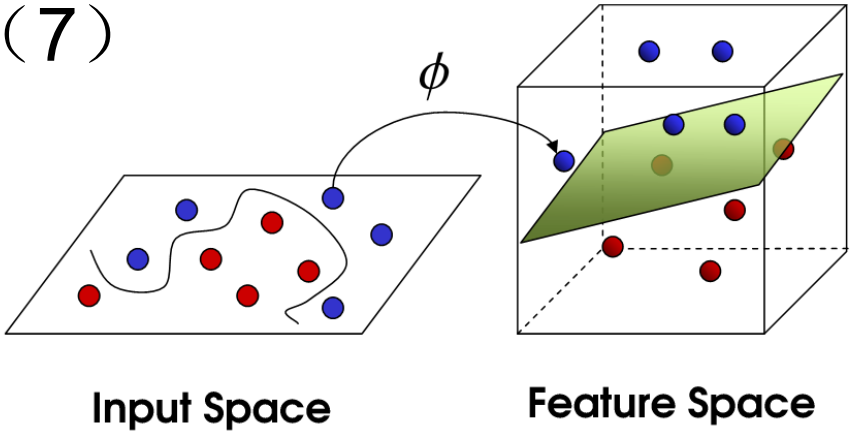
(5)



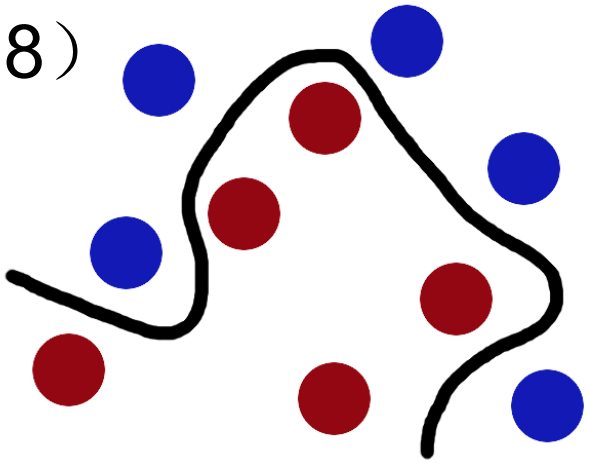
(6)



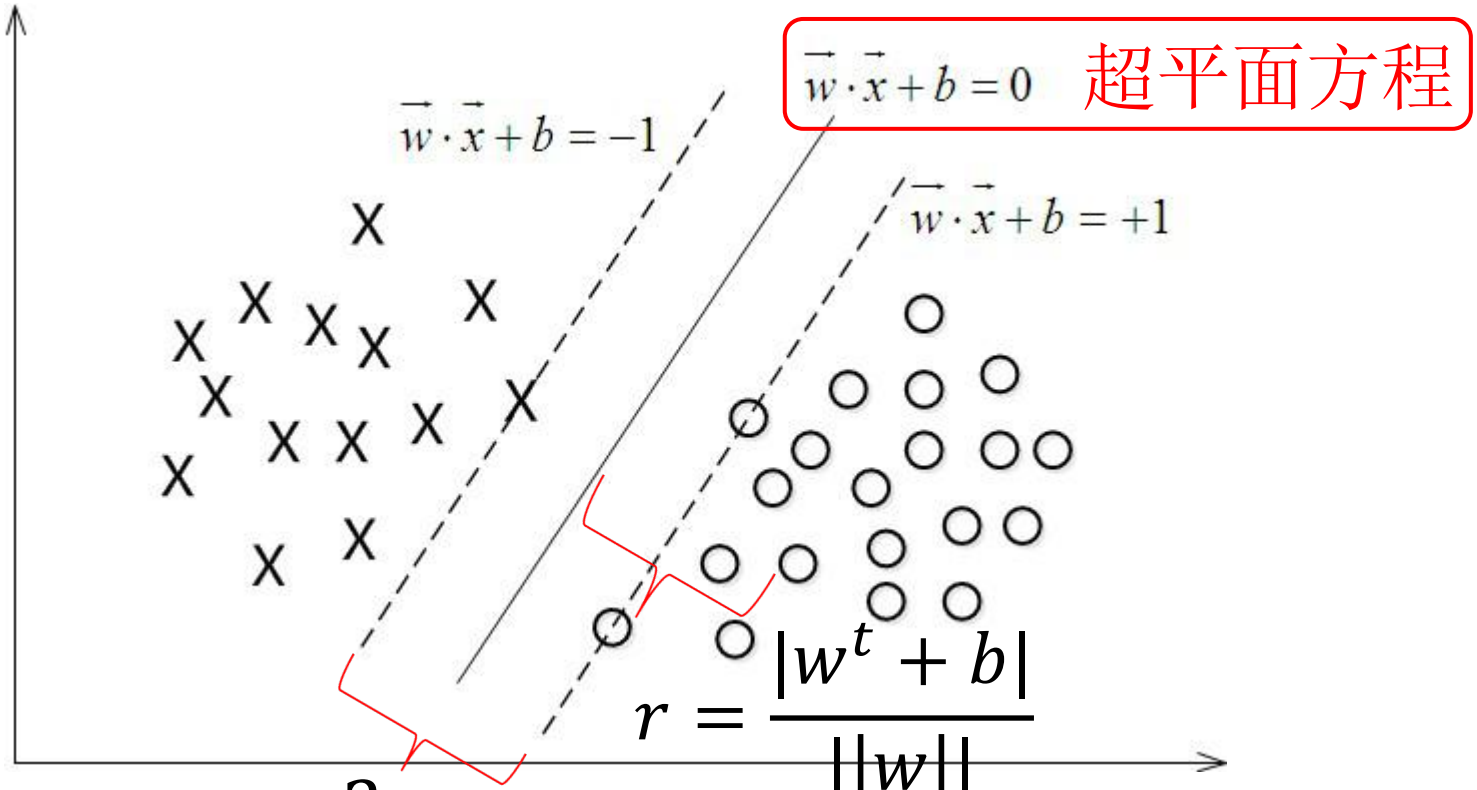
(7)




(8)



# SVM


$\vec{w}\cdot\vec{x}+b=0$ 超平面方程
$\vec{w}\cdot\vec{x}+b=-1$
$\vec{w}\cdot\vec{x}+b=+1$
$r=\frac{|w^t+b|}{||w||}$
$\gamma=\frac{2}{||w||}$ 间隔

# SVM



- 最大间隔：寻找参数$w$和$b$，使得$\gamma$最大

$$\max \frac{1}{\|w\|} \quad s.t., y_i(w^T x_i + b) \geq 1, i = 1, \ldots, n$$

- 等价于：$\min \frac{1}{2} \|w\|^2 \quad s.t., y_i(w^T x_i + b) \geq 1, i = 1, \ldots, n$
- 凸二次规划问题，能用优化计算包求解，但可以有更高效的办法。

# SVM



- 对偶问题
  - 拉格朗日乘子法
  - 最终模型：$f(x) = w^T x + b = \sum_{i=1}^{m} \alpha_i y_i x_i^T x + b$
- 若不存在一个能正确划分两类样本的超平面，怎么办？
  - 将样本从原始空间映射到一个更高维的特征空间，使样本在这个特征空间内线性可分。
  - 预测：$f(x) = w^T \emptyset(x) + b = \sum_{i=1}^{m} \alpha_i y_i \emptyset(x_i)^T \emptyset(x) + b$
  - 核函数：$k(x_i, x_j) = \emptyset(x_i)^T \emptyset(x_j)$


# 如何使用SVM?



- 入门级：实现并使用各种版本的SVM
- 专业级：尝试、组合核函数
- 专家级：根据问题而设计目标函数、替代损失、进而...


# SVM的应用



- 有助于文本和超文本分类
- 图像的分类
- 识别手写字符
- 生物科学和其他科学领域，比如对高达90%正确分类的化合物进行蛋白质分类。
- ……


# SVM常用软件包



- LIBSVM (by台湾大学林智仁教授)
  - https://www.csie.ntu.edu.tw/~cjlin/libsvm/
- Matlab
  - SVM and Kernel Methods Matlab Toolbox
- R
  - e1071 - Machine learning library for R
- Python
  - numpy
  - scipy

# 应用案例



- 基于MapReduce的大规模短文本分类算法
  - 提供了1万条已经标注出所属类别的短文本样本数据作为训练样本，一共有480个类别。
  - 测试数据有1000万条查询短文本样本数据，其中有少数不属于这480类的异类测试样本，需要对这些大量的短文本进行分类，并能标识出不属于以上480类的异类样本。
  - 每个短文本样本数据由一个n维的高维特征向量构成。


# 应用案例



- 特点：
  - 短文本包含的词语相对较少，因此其构造出来的特征向量往往具有高维稀疏性；
  - 短文本的目标分类种类很多，因此不再是一个简单的二分类问题，而直接采用一个多类别分类起可获得的精度较差，且其训练流程难以并行化。
- 解决方案：
  - SVM针对高维空间数据的分类处理效果较好，尤其Linear SVM分类器的处理速度较快
  - 因此，使用二类别分类器的Linear SVM作为基本分类器

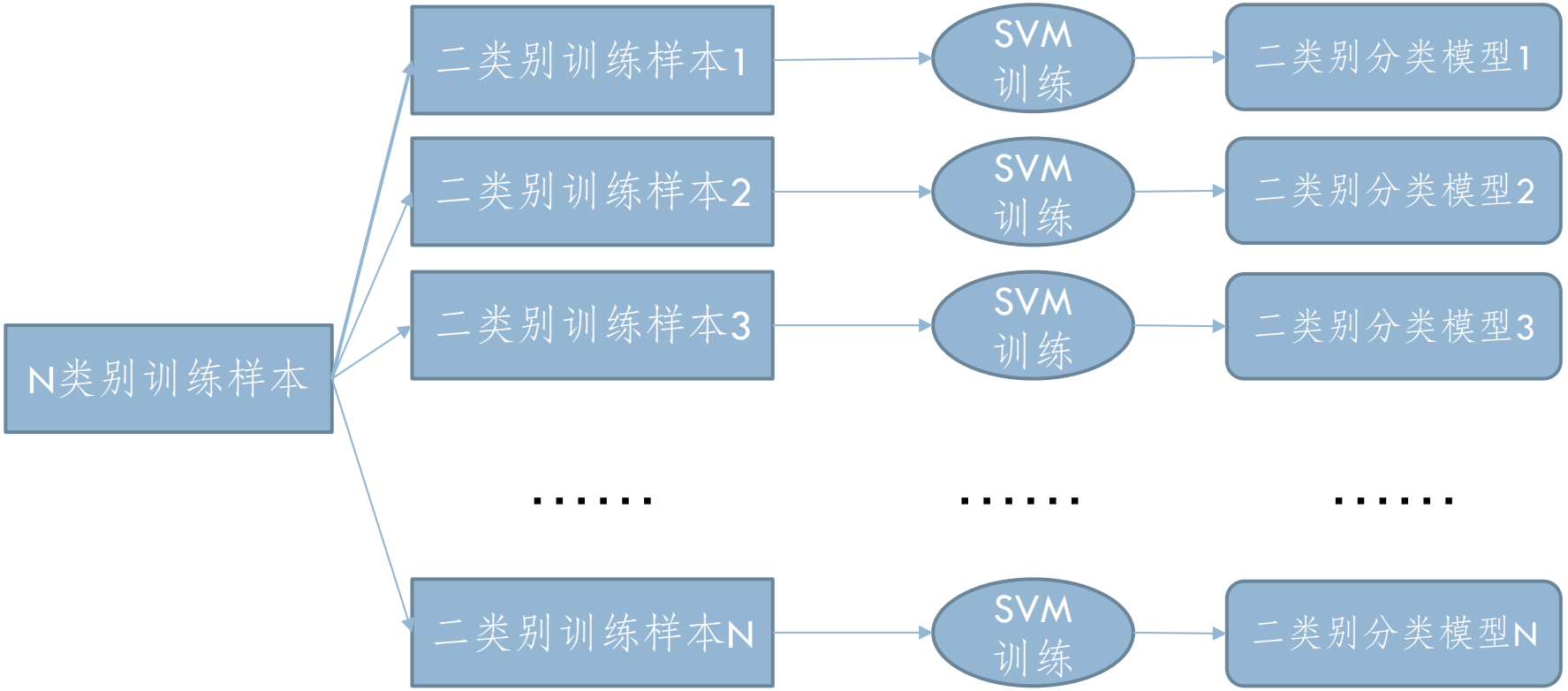

# 基本算法思路



- 训练阶段，对于多类（480 类）问题，为了提高分类精度，首先针对每个类做一个2-Class两类分类器；
- 预测阶段，分别用480 个分类器对每个待预测的样本进行分类并打分，选择分类为“是”且打分最高的类别作为该样本可能的预测类别；如打分低于最低阈值，则将该测试样本判定为不属于480类的异类。

# 流程图

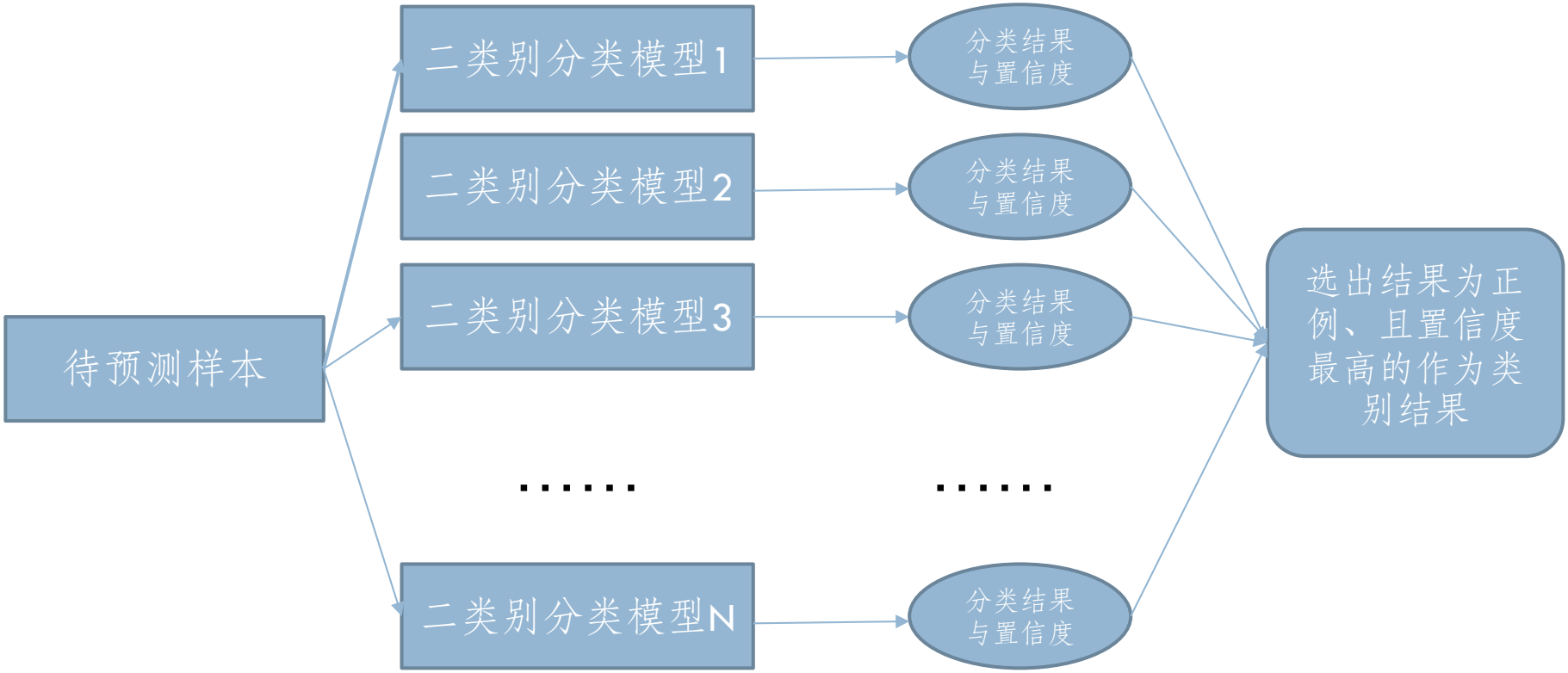





N类训练样本训练出N个二类别分类器的流程图

# 流程图






对待预测样本确定其类别的流程图


# 并行化分类训练算法设计实现



- 训练样本很大，样本类别多，待预测的样本数量巨大等问题会导致训练时间很长。
- 实现基于MapReduce并行化的多分类器，包括：
  - 数据预处理
  - 训练N个二类别分类器
  - 预测样本

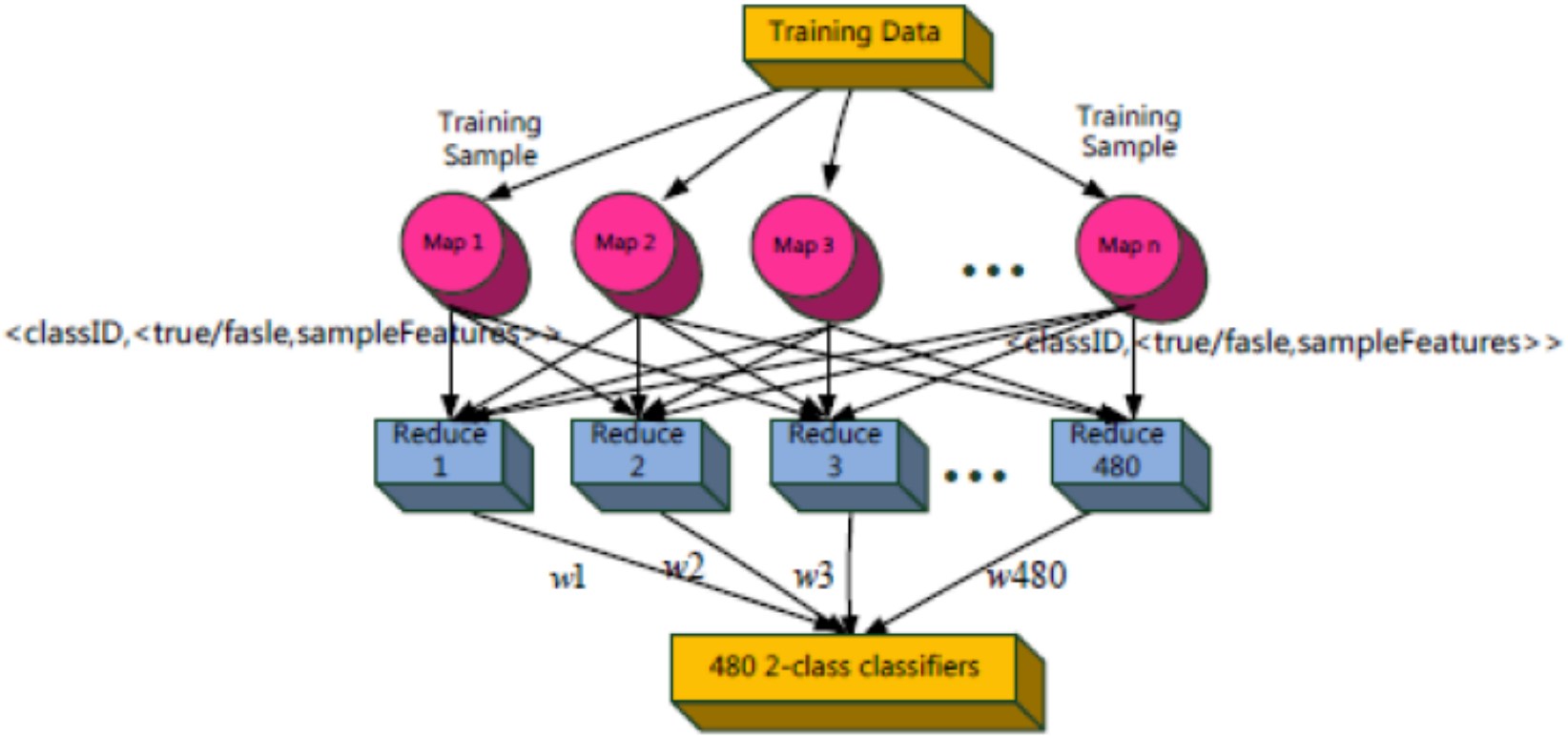
# 基本算法设计思路



- 第一步：用训练数据产生480个2-class分类器模型
  - Map：将每个训练样本的分类标签ClassID作为主键，输出(ClassID, <true/false，特征向量>)
  - Reduce: 具有相同ClassID的键值对进入同一个Reduce，然后训练出一个2-Class SVM分类模型共输出480个2-Class SVM分类模型

# 基本算法设计思路






基于MapReduce的480个两类分类器并行训练算法

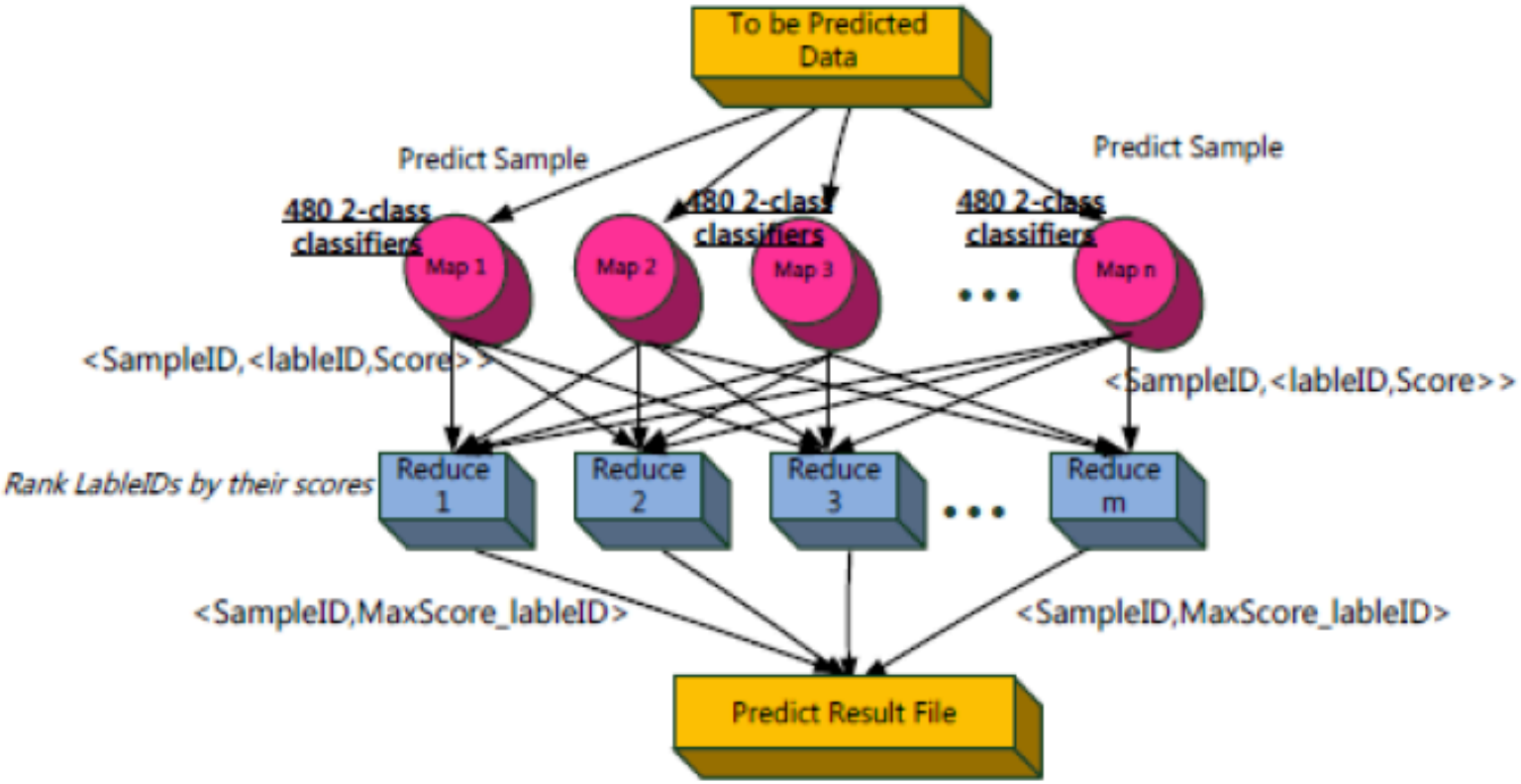

# 基本算法设计思路



- 第二步：用480个2-Class分类器模型处理测试数据
  - Map：将每个测试样本，以SampleID作为主键，输出(SampleID, <LableID，评分Score>)
  - Reduce: 具有相同SampleID的键值对进入同一个Reduce，然后以最高评分者对应的标记作为该样本最终的标记；虽然是最高评分，但仍然低于最小阈值，则判定为不属于已知的480个类

# 基本算法设计思路






基于MapReduce的480个两类分类器并行预测算法