

MapReduce 数据挖掘基础算法(II)

分类算法并行化



摘要

2

- 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



聚类

3

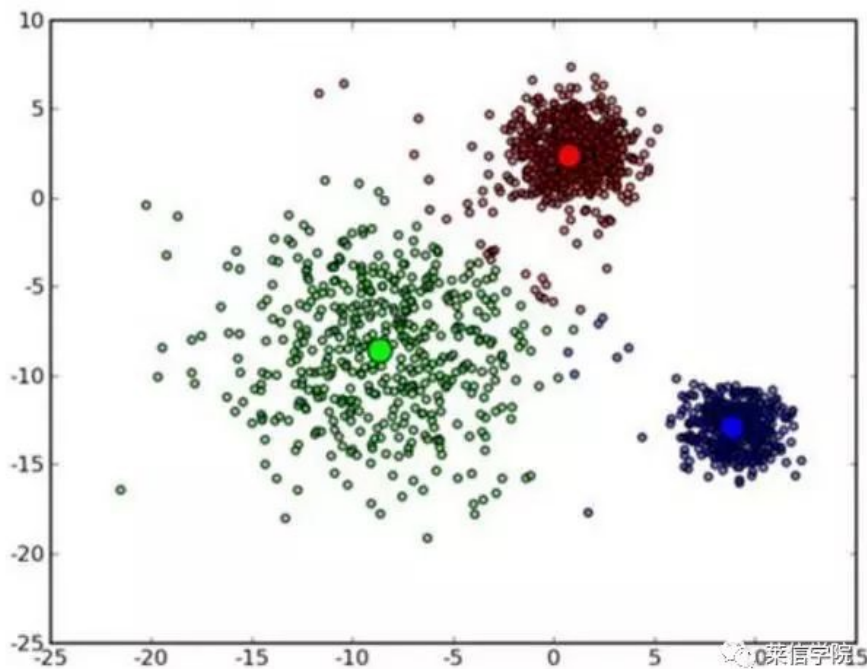
- 聚类与分类的不同在于，聚类所要求划分的类是**未知**的。聚类的目的是使得属于同类别的对象之间的差别尽可能的小，而不同类别上的对象的差别尽可能的大。因此，聚类的意义就在于将观察到的内容依据相应算法组织成类分层结构，把类似的事物组织在一起。
- 在机器学习中，聚类是一种无监督学习。也就是说，聚类是在预先不知道欲划分类别的情况下，根据信息相似度原则进行信息聚类的一种方法。通过聚类，人们能够识别密集的和稀疏的区域，因而发现全局的分布模式，以及数据属性之间的有趣的关系。



聚类

4

- 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。





分类

5

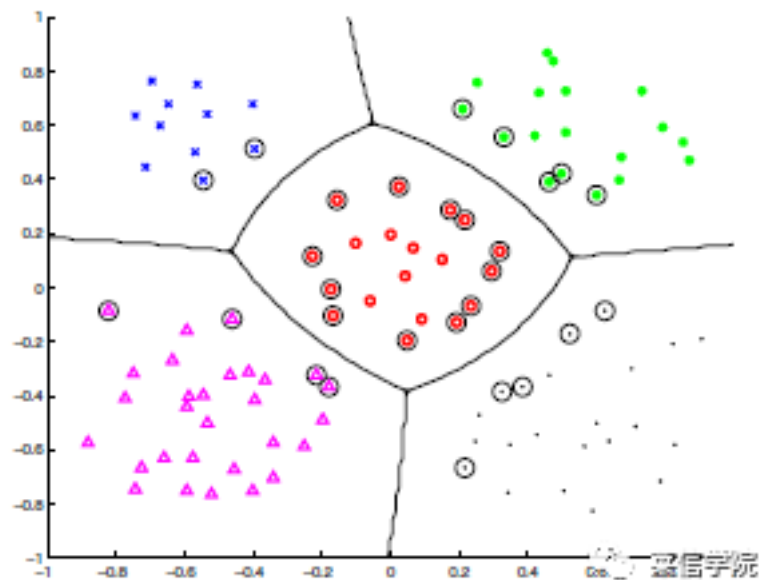
- 分类（**Classification**）是机器学习和数据挖掘中最重要、最频繁使用的算法。
- 分类算法的基本作用是：从一组已经带有分类标记的训练样本数据集来预测一个测试样本的分类结果。
- 从机器学习的观点，分类属于监督学习，每个训练样本的数据对象已经有类标识，通过学习可以形成表达数据对象与类标识间对应的知识。
- 分类具有广泛的应用，例如医疗诊断、信用卡的信用分级、图像模式识别、营销用户画像等。



分类

6

- 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

7

□ 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

8

- 分类算法的基本描述：
 - ▣ 一个训练数据集 $TR = \{tr1, tr2, tr3, \dots\}$
 - ▣ 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： $A = \{a1, a2, a3, \dots\}$ ，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

9

- 对于一个测试样本数据集 $TS = \{ts1, ts2, ts3, \dots\}$, 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

10

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 |
| ... | ... | ... | ... | ... | ... |



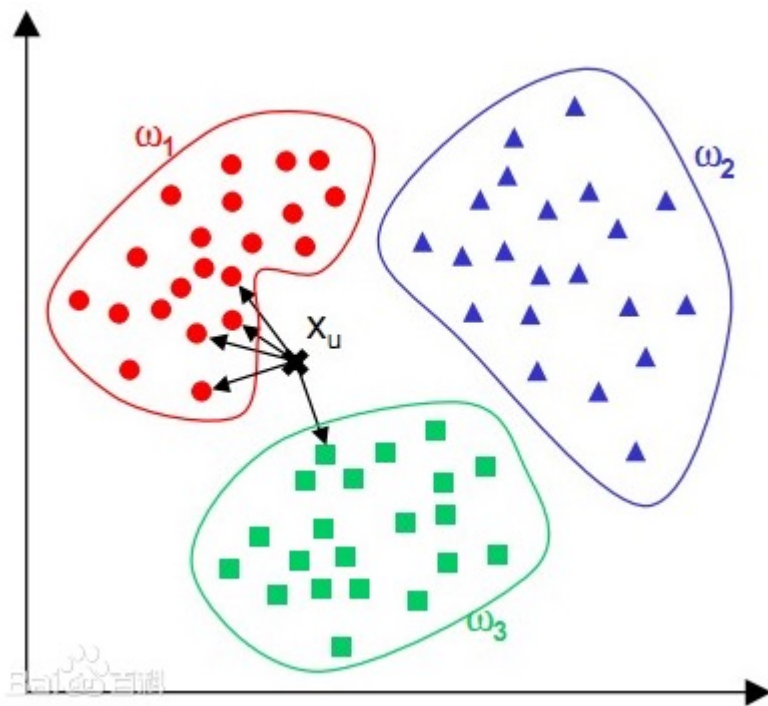
测试样本数据

| 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

11





K-最近邻(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 $y' = \sum Si * yi / \sum Si$, $k=[0, k-1]$, 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 $y' = \sum Si * yi / \sum si$ 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
```

```
         $\Phi \rightarrow \text{MaxS}(k)$ 
```

```
         $ts \rightarrow \text{tsid}, A', y'$ 
```

```
        for i=0 to TR.length {
```

```
             $\text{TR}[i] \rightarrow \text{trid}, A, y$ 
```

```
             $S = \text{Sim}(A, A')$ ;
```

```
            若S属于k个最大者,  $(S, y) \rightarrow \text{MaxS}$ ;
```

```
        }
```

```
        根据MaxS和带加权投票表决模型计算出  $y' = \sum S_i * y_i / \sum S_i$ 
```

```
        emit(tsid, y')
```

```
    }
```

```
}
```



分类算法的评估

15

| | | 真实值 | |
|-----|----------|----------|----------|
| | | Positive | Negative |
| 预测值 | Positive | TP（真阳性） | FP（假阳性） |
| | Negative | FN（假阴性） | TN（真阴性） |

第一个字母表示真实值与预测值划分正确与否，**T**表示判定正确(True)，**F**表示判定错误(False)。

第二个字母表示分类器判定结果(预测结果)，**P**表示判定为正例，**N**表示判定为负例。



分类算法的评估

16

□ 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

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} = \frac{\text{正确预测的样本数}}{\text{所有的样本数}}$$

□ 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

$$Precision = \frac{TP}{TP + FP} = \frac{\text{分类正确的正样本个数}}{\text{被分类器判定为正样本的样本个数}}$$



分类算法的评估

17

□ 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。

$$Recall = \frac{TP}{TP + FN} = \frac{\text{分类正确的正样本个数}}{\text{真正的正样本个数}}$$

□ F1 Score: 精确率和召回率的调和均值

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



分类算法的评估方法

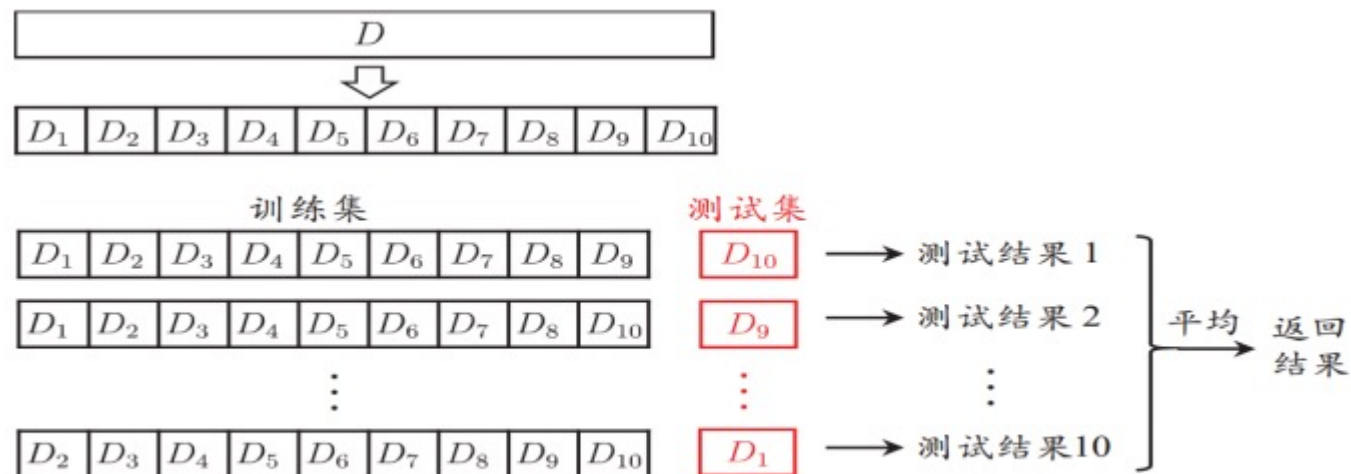
18

- 通常将包含 m 个样本的数据集 $D=\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_m, y_m)\}$ 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

19



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

20

- 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

21

□ 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： $X = \{x_1, x_2, \dots, x_n\}$ ，假定有 m 个类，分别用 Y_1, Y_2, \dots, Y_m 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 Y_i ，则一定有 $P(Y_i|X) > P(Y_j|X)$, $1 \leq j \leq m, j \neq i$
- 根据贝叶斯定理

$$P(Y_i|X) = \frac{P(Y_i) * P(X|Y_i)}{\sum_{i=1}^k P(Y_i) * P(X|Y_i)} = \frac{P(Y_i) * P(X|Y_i)}{P(X)}$$

- 由于 $P(X)$ 对于所有类为常数，概率 $P(Y_i|X)$ 可转化为概率 $P(X|Y_i) * P(Y_i)$ 。



朴素贝叶斯分类算法

22

- 如果训练数据集中有很多具有相关性的属性，计算 $P(X|Y_i)$ 将非常复杂，为此，通常假设各属性是互相独立的，这样 $P(X|Y_i)$ 的计算可简化为求 $P(x_1|Y_i)$ ， $P(x_2|Y_i)$ ， \dots ， $P(x_n|Y_i)$ 之积；而每个 $P(x_j|Y_i)$ 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 Y_i 的概率 $P(X|Y_i) * P(Y_i)$ ，然后选择其中概率最大的 Y_i 作为其类别。



朴素贝叶斯分类并行算法

23

- 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 Y_i 的核心任务成为：根据训练数据集计算 Y_i 出现的频度和所有属性值 x_j 在 Y_i 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 Y_i 出现的频度 F_{Y_i} (即 $P(Y_i)$)、以及每个属性值出现在 Y_i 中的频度 $F_{x_j|Y_i}$ (即 $P(x_j|Y_i)$)



朴素贝叶斯分类并行算法

24

- 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 Y_i 和每个 x_j 在 Y_i 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 x_j ，根据从训练数据集计算出的 $F_{xY_{ij}}$ 进行求积得到 $F_{xY_{ij}}$ (即 $P(X|Y_i)$)，再乘以 F_{Y_i} 即可得到 X 在各个 Y_i 中出现的频度 $P(X|Y_i) * P(Y_i)$ ，取得最大频度的 Y_i 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

25

□ 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

27

□ 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

28

```
map(key, ts){ // ts为一个测试样本
    ts  $\rightarrow$  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 $P(X|Y_i)$ 
        if(FXYi* FYi>MaxF) { MaxF= FXYi*FYi; idx= i; }
    }
    emit(tsid, FY[idx].Yi)
}
```

决策树

29

- 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。

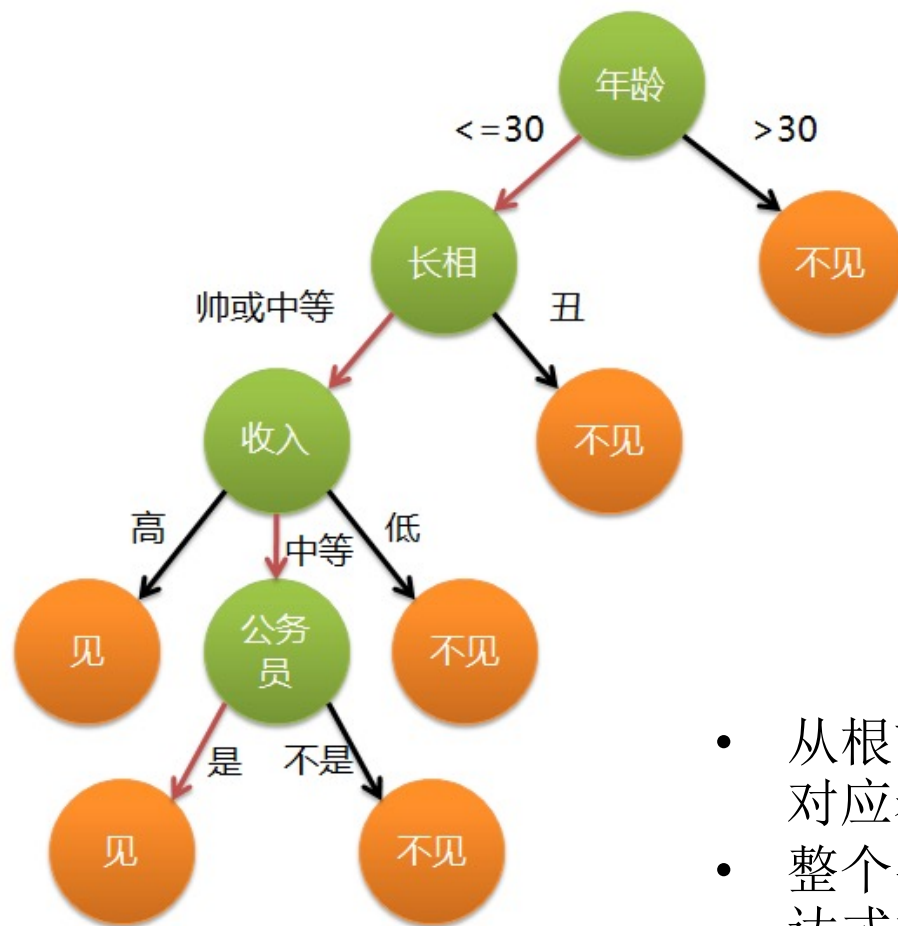


引自：机器学习，周志华，清华大学出版社，2016年1月



决策树

30



- 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

31

- 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

32

- 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 $\text{TreeGenerate}(D, A)$

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

利用当前结点的后验分布

5: if $A = \emptyset$ 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_* ;

决策树算法的核心

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 $\text{TreeGenerate}(D_v, A \setminus \{a_*\})$ 为分支结点

15: end if

16: end for

将父结点的样本分布作为当前结点的先验分布

输出：以 node 为根节点的一棵决策树



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。

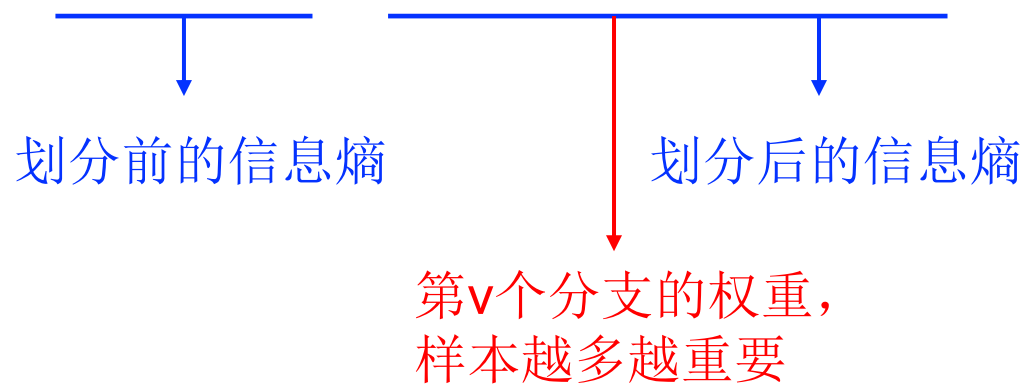


信息增益

35

- 离散属性 a 的取值: $\{a^1, a^2, \dots, a^v\}$
- 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:

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



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：

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

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

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。



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：

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

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

38

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

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

| 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'es and 3 no's. Hence

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

Similarly,

$$Gain(income) = 0.029$$

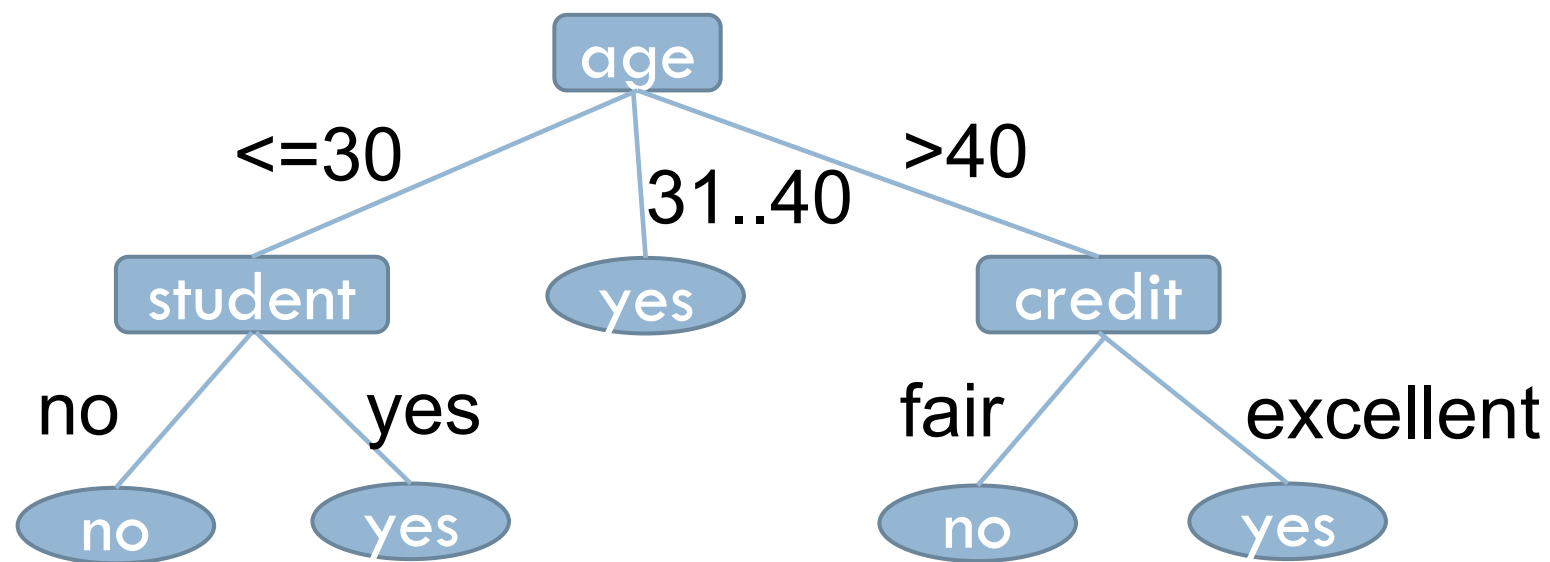
$$Gain(student) = 0.151$$

$$Gain(credit_rating) = 0.048$$



Example

39





更多问题

40

- 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

41

□ 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

42

- 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

44

- 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。

THANK YOU



南京大學
NANJING UNIVERSITY

南京大学计算机软件研究所
Institute of Computer Software, Nanjing University