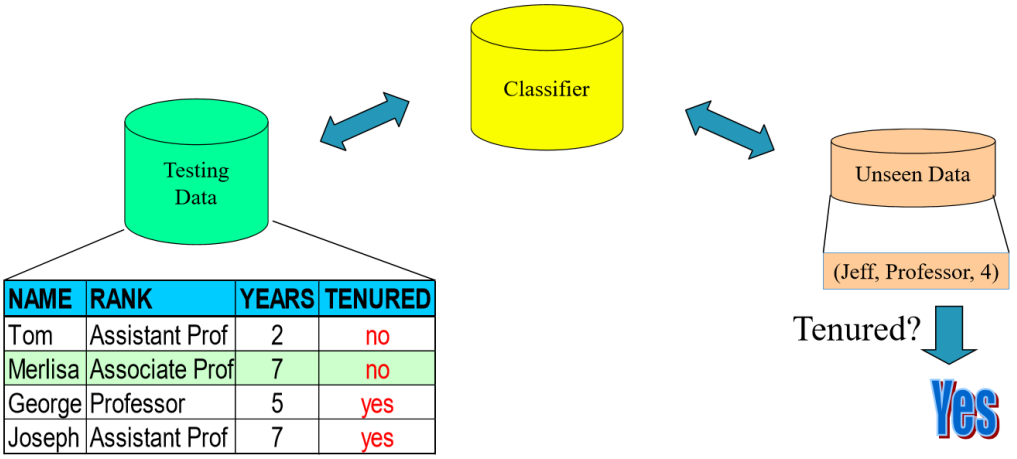
分类的定义?

- 分类指通过分析一个类别已知的数据集（训练集）的特征来建立一组模型，该模型可用以预测类别未知的数据项（测试集）的**类别**。（常用分类树）
- 分类的目的是获得一个分类函数或分类模型（也常常称作分类器），该模型能把数据库中的数据项映射到某一个给定类别。
- 分类可用于提取描述重要数据类的模型或预测未来的数据趋势。
    - 分类是预测分类（离散、无序的）标号
    - 预测是建立连续值函数模型

分类实现的原理?

- 构建模型：预设分类类别
    - 对每个样本进行类别标记
    - 训练集构成分类模型
    - 分类模型可表示为：分类规则、决策树或数学公式。
- 使用模型：识别未知对象的所属类别
    - 模型正确性的评价
        - 已标记分类的测试样本与模型的实际分类结果进行比较
        - 模型的正确率是指测试集中被正确分类的样本数与样本总数的百分比。测试集与训练集相分离，否则将出现过拟合（over-fitting）现象。
-



NAME	RANK	YEARS	TENURED
Tom	Assistant Prof	2	no
Merlisa	Associate Prof	7	no
George	Professor	5	yes
Joseph	Assistant Prof	7	yes

典型算法有哪些?

- **K-NN分类（K最近邻法）**
  - 最初由Cover和Hart于1968年提出的，是一个理论上比较成熟的方法。
  - 该方法的思路非常简单直观：**如果一个样本在特征空间中的k个最相似（即特征空间中最邻近）样本中的大多数属于某一个类别， 则该样本也属于这个类别。**
  - 该方法在分类决策上只依据最邻近的一个或者几个样本的类别来决定待分类样本所属的类别。
  - 该算法较适用于**样本容量比较大的类域的自动分类**，而那些样本容量较小的类域采用这种算法比较容易产生误分。
- **SVM分类方法**
  - 即支持向量机（Support Vector Machine）法，由Vapnik等人于1995年提出，具有相对优良的性能指标。
  - 该方法是建立在统计学习理论基础上的机器学习方法。通过学习，SVM可以自动寻找出那些对分类有较好区分能力的支持向量，由此构造出的分类器可以**最大化类与类的间隔**，因而有较好的适应能力和较高的分准率。
  - 该方法只需要由各类域的边界样本的类别来决定最后的分类结果。
  - SVM法对**小样本情况下的自动分类**有着较好的分类结果。
  - 例子：

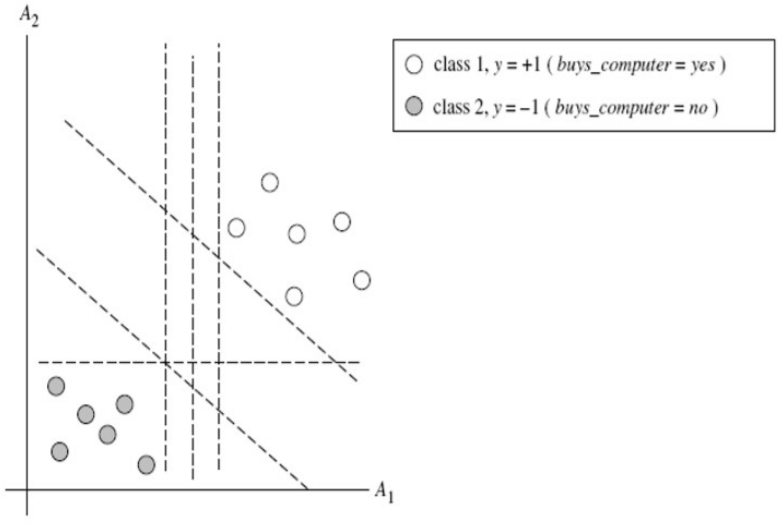


Figure 6.20 The 2-D training data are linearly separable. There are an infinite number of (possible) separating hyperplanes or "decision boundaries." Which one is best?

- **决策树算法**
  - 决策树归纳是一种经典的分类算法。
  - 它采用**自顶向下、递归的、各个击破**的方式构造决策树。树的每一个结点上使用信息增益度量选择属性，可以从所生成的决策树中提取出分类规则。
  - 决策树分类是用属性值对样本集逐级划分，直到一个节点仅含有同一类的样本为止。
  - 决策树首先起源于Hunt等人提出的概念学习系统（Concept Learning System,CLS），然后发展到**Quinlan的ID3算法**，最后演化为能处理连续属性值的**C4.5**算法。

形式：决策树

- **决策树输入**
  - 一组带有类别标记的样本
- **决策树输出**
  - 一颗二叉或多叉树
  - 二叉树的内部节点（非叶子节点）一般表示为一个逻辑判断，如形式为(ai=vi)的逻辑判断， 其中ai是属性，vi是该属性的某个属性值；树的边是逻辑判断的分支结果。
  - 多叉树（ID3）的内部节点是属性，边是该属性的所有取值，有几个属性值，就有几条边。 树的叶子节点则是类别标记。
  - 例如

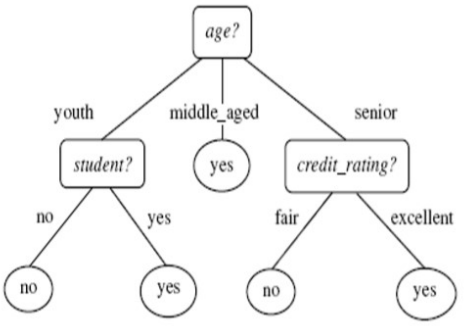


Figure 6.2 A decision tree for the concept *buys_computer*, indicating whether a customer at *AllElectronics* is likely to purchase a computer. Each internal (nonleaf) node represents a test on an attribute. Each leaf node represents a class (either *buys_computer* = *yes* or *buys_computer* = *no*).

■ 根据加薪百分比、工作时长、法定节假日、及医疗保险三个属性来判断一个企业的福利状况。

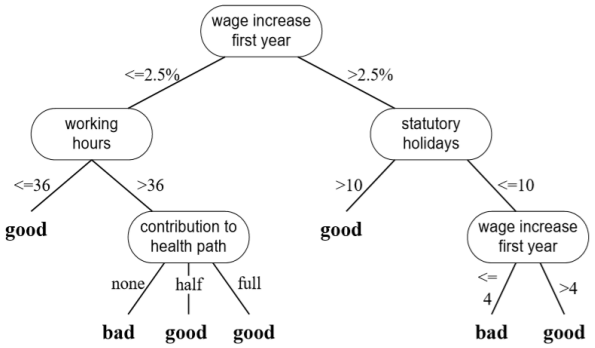
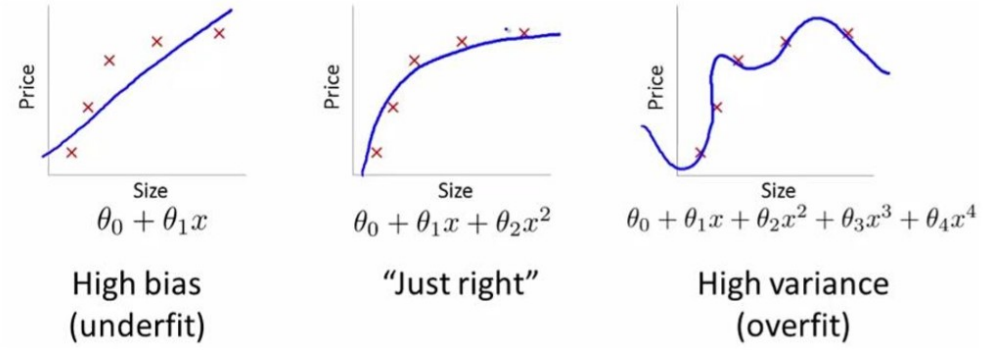


图　　公司福利条件决策树示例

- **构造决策树方法**
  - 采用**自上而下**的递归构造
    - 如果训练样本集中所有样本是同类的，则将它作为叶子节点，节点内容即是该类别标记；
    - 否则，根据某种策略选择一个属性，按照属性的不同取值，将样本集划分为若干子集，使得每个子集上的所有样本在该属性上具有同样的属性值。
    - 然后再依次处理各个子集。
  - 实际上是“**分而治之**”的策略。
- **构造决策树条件**
  - 构造好的决策树的关键是：如何选择好的逻辑判断或属性
  - 对于同样一组样本，可以有很多决策树能符合这组样本。
    - 研究表明，一般情况下，树越小则树的预测能力越强。
    - 要构造尽可能小的决策树，关键在于选择恰当的逻辑判断或属性。
    - 由于构造最小的树是NP问题，因此只能采用启发式策略选择好的逻辑判断或属性。
- 由于训练数据不完美，需要**克服噪声**和**决策树剪枝（克服噪声的技术，也能使树简化）**。

过拟合是什么?

- 基本的决策树构造算法没有考虑噪声，生成的决策树完全与训练样本拟合。

- 在有噪声的情况下，完全拟合将导致**过分拟合（overfitting）**，即对训练数据的完全拟合反而不具有很好的预测性能。
- 

剪枝技术是什么?

- **剪枝技术**是一种克服噪声的技术，同时它也能使树得到简化而变得更容易理解。
  - 向前剪枝：在生成树的同时决定是继续对不纯的训练子集进行划分还是停机。
  - 向后剪枝：是一种两阶段法：拟合 - 化简 （fitting-and-simplifying），首先生成与训练数据完全拟合的一棵决策树，然后从树的叶子开始剪枝，逐步向根的方向剪。
  - 例

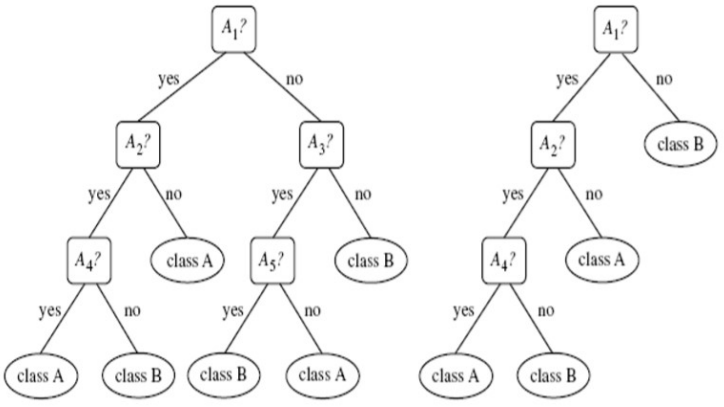


Figure 6.6 An unpruned decision tree and a pruned version of it.

- **剪枝的局限性**
  - 剪枝并不是对所有的数据集都好，就像最小树并不是最好（具有最大的预测率）的树。
  - 当数据稀疏时，要**防止过分剪枝（over-pruning）**。
  - 从某种意义上而言，剪枝也是一种偏向（bias），对有些数据效果好而有些数据则效果差。

ID3怎么做，什么标准选属性？

- Quinlan提出的**ID3算法**，对CLS算法做出了改进。它的基本算法仍然来自于CLS，但**使用熵来选择属性**，效果非常理想。
- ID3使用**信息增益**作为属性选择度量，设节点N代表或存放划分D的元组，选择具有最高信息增益的属性作为节点N的分裂属性。该属性使结果划分中的元组分类所需的信息量最小。
- **信息增益**
    - 对D中的元组分类所需的期望信息由下式给出：

$$info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i)$$

    - 其中，p_i是D中任意元组属于类c_i的概率，并用$|c_i, D|/|D|$估计。使用以2为底的对数函数，因为信息用二进位编码。 $Info(D)$是识别D中元组的类标号所需要的平均信息量。注意，我们这里所具有的信息只是每个类的元组所占百分比，$Info(D)$又称D的熵。
    - 假设我们要按属性A划分D中的元组，其中属性A根据训练数据的观测具有v个不同的值$\{a_1, a_2, .., a_v\}$
    - 为了得到准确的分类我们还需要多少信息？这个量由下式度量：

$$info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times info(D_j)$$

> 其中，项$|D_j|/|D|$充当第j个划分的权重。$info_A(D)$是基于按A划分对D的元组分类所需要的期望信息，还需要的期望信息越小，划分的纯度越高。

    - 信息增益定义为原来的信息需求(即仅基于类比例)与新的需求(即对A划分之后得到的)之间的差，即：

$$Gain(A) = Info(D) - info_A(D)$$

    - 换言之，$Gain(A)$告诉我们通过A的划分我们得到了多少，它是知道A的值而导致的信息需求的期望减少。选择具有最高信息增益$Gain(A)$的属性A作为节点N的分裂属性。这等价于按能做“最佳分类”的属性A划分，使得完成元组分类还需要的信息最小。
- **算法举例：**

Table 6.1 Class-labeled training tuples from the *AllElectronics* customer database.

RID	age	income	student	credit_rating	Class: buys_computer
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle_aged	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle_aged	low	yes	excellent	yes
8	youth	medium	no	fair	no
9	youth	low	yes	fair	yes
10	senior	medium	yes	fair	yes
11	youth	medium	yes	excellent	yes
12	middle_aged	medium	no	excellent	yes
13	middle_aged	high	yes	fair	yes
14	senior	medium	no	excellent	no

- $info(D) = -\frac{9}{14}\log_2\frac{9}{14} - \frac{5}{14}\log_2\frac{5}{14} = 0.940$
- $info_{age}(D) = \frac{5}{14}\left(-\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5}\right) + \frac{4}{14}\left(-\frac{4}{4}\log_2\frac{4}{4} - \frac{0}{4}\log_2\frac{0}{4}\right) + \frac{5}{14}\left(-\frac{3}{5}\log_2\frac{3}{5} -$

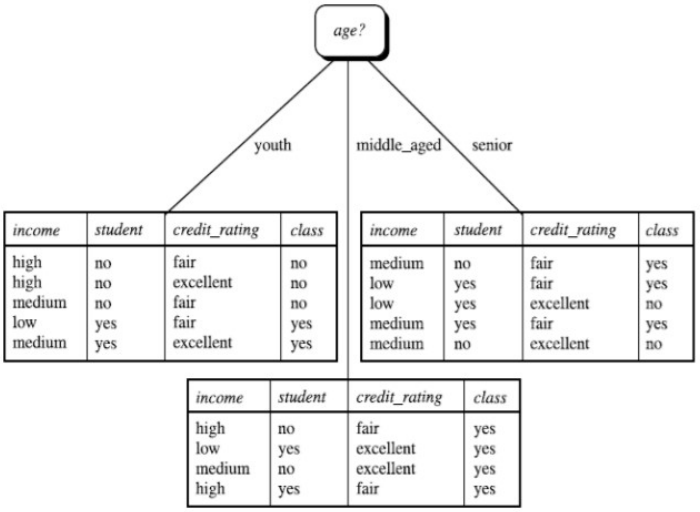


Figure 6.5 The attribute *age* has the highest information gain and therefore becomes the splitting attribute at the root node of the decision tree. Branches are grown for each outcome of *age*. The tuples are shown partitioned accordingly.

- **ID3算法的优势与不足**
  - **优势：**
    - 算法理论清晰
    - 方法简单
    - 学习能力较强
  - **不足：**
    - 对较小的数据集有效

        - 对噪声比较敏感
        - 当数据集增大时，决策树可能会改变

分类的评判标准?

- 预测的正确性
    - 混淆矩阵

        - 准确率：$\frac{TP+TN}{P+N}$
        - 错误率：$\frac{FP+FN}{P+N}$
        - 召回率：$\frac{TP}{P}$
        - 精度：$\frac{TP}{TP+FP}$
    -
        - F1分数：$\frac{2\times precision\times recall}{precision+recall}$

预测的类

实际的类	yes	no	合计
yes	TP	FN	P
no	FP	TN	N
合计	P'	N'	$P+N$

- 时间
    - 构建模型的时间
    - 使用模型所需的时间
- 健壮性
    - 处理噪声及缺失值的能力
- 可扩展性
- 可操作性
- 规则的优化
    - 决策树的大小
    - 分类规则的简洁性
- 交叉验证
    - 交叉验证的基本思想
        - 数据分组：把在某种意义下将原始数据(dataset)进行分组，一部分做为训练集(train set)，另一部分做为验证集(validation set or test set)。
        - 模型训练：首先用训练集对分类器进行训练，再利用验证集来测试训练得到的模型(model)，以此来做为评价分类器的性能指标。

- 常见交叉验证模式
  - Holdout验证
  - **K-fold corss-validation**
  - 留一验证

- ROC曲线
  - ROC曲线是衡量分类模型效果的最重要的图形展现形式
  - 绿色折线是**理想曲线（Ideal Curve）**，代表目标类别完全识别，无一漏网也无一错认
  - 红色直线是**随机曲线（Random Curve）**，代表目标类别完全无法识别，随机检测
  - 蓝色曲线是**性能曲线（Performance Curve）**，代表目标类别被模型识别效果，**越接近绿线性能越好**

- ROC曲线是用来评判当前模型效果的一个指标
  - 混淆矩阵是算准确率的，不一样。
  - ROC等于几：提升曲线和随机曲线包围起来的面积，越大越好。

额外内容

C4.5算法

- C4.5算法
  - ID3有很多改进算法，其中Quinlan在1994年开发出的C4.5算法流行最广。
- C4.5的改进主要体现在两方面：
  - 解决了**连续数据值**的学习问题；
  - 提供了将**学习结果决策树到等价规则集的转换**功能。

神经网络算法

- 人工神经网络（Artificial Neural Network，ANN）是20世纪80年代后期迅速发展起来的人工智能技术。
- 算法对**噪声数据具有很高的承受能力**，对未经训练的数据具有分类模拟的能力，因此在网站信息、生物信息和基因以及文本的数据挖掘等领域得到了越来越广泛的应用。
- 在多种ANN模型中，**反向传播（Back Propagation，BP）网络**是应用最广的一种。

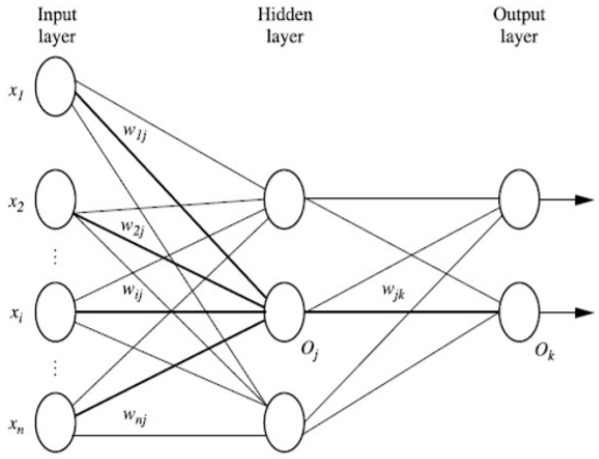
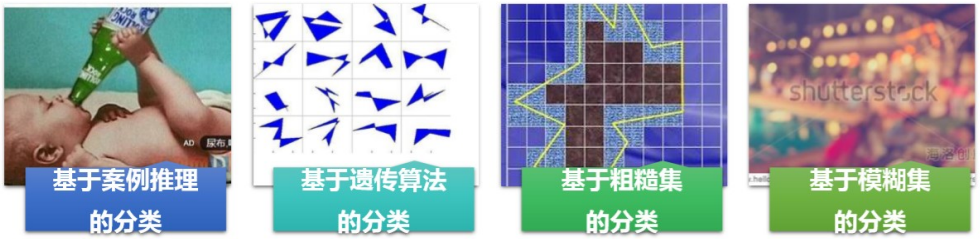
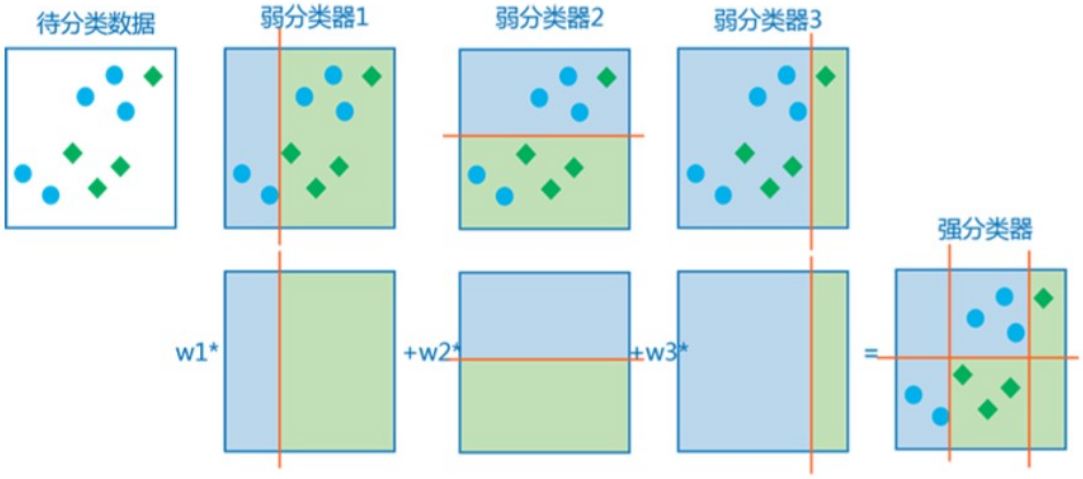


Figure 6.15 A multilayer feed-forward neural network.

其它的分类算法



组合分类（代表算法有随机森林；Boosting和GBDT三种）

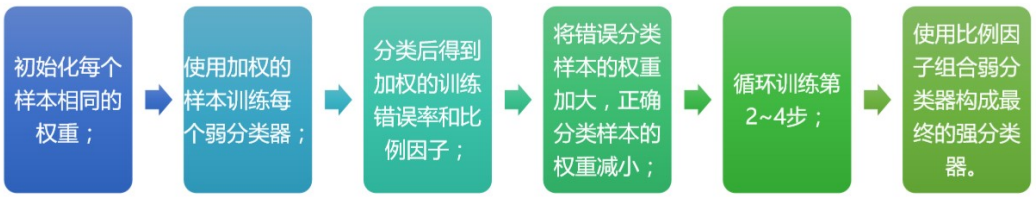


随机森林

- **随机森林，用随机的方式建立一个森林。**
- 森林里面有很多的决策树组成，随机森林的**每一棵决策树之间是没有关联的。**
- “**投票分类**”：当有一个新的输入样本进入的时候，就让森林中的每一棵决策树分别判断该样本应属哪一类，然后看看哪一类被选择最多，就预测该样本为那一类。
- 随机森林算法比喻说法：
  - 每一棵决策树就是一个精通于某一个**窄领域的研究员**；
  - 在随机森林中就有了很多个**精通不同领域的研究员**；
  - 对一个新的问题，可以用不同的角度去看待它，最终由所有研究员**投票**得到结果。

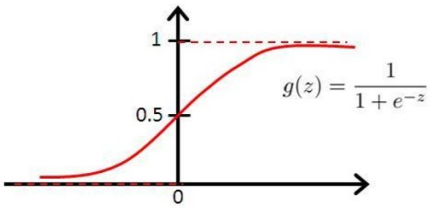
Boosting

- Boosting：提升、促进。
- 一般Boosting算法都是一个**迭代**的过程，每一次新的训练都是为了**改进上一次的结果。**
- 算法流程：



GBDT

- Gradient：梯度、倾斜度。
- 与传统的Boost的区别：**每一次的计算是为了减少上一次的残差(residual)**。为了消除残差，算法在残差减少的梯度(Gradient)方向上建立一个新的模型。
  - 传统Boost：对正确、错误的样本进行加权。
  - GBDT：新模型建立是为了使之前模型的残差往梯度方向减少。



大数据分类挑战

- 单一学习变为分布式学习
  - 传统的分类挖掘方法以单一学习样本集为基础，而大数据的分布式收集特性决定分类学习需要分布式进行，因而对应的分布式学习策略和方法需要研究。
- 静态数据变为动态数据
  - 动态流动的流式大数据和传统数据库存储的静态数据有显著的不同，不可能一次性将所有数据存储起来再进行离线式的挖掘，必须探索在线实时的收集技术和随时间变化的增量式的挖掘方法。
- 分布式很难保证学习样本的纯度要求
  - 传统的分类挖掘技术对学习样本集要求较高，而分布式、流式大数据的分类挖掘需要多节点、多步骤协同处理，很难保证学习样本集的纯度，所以必须针对这类大数据的挖掘特点来探索鲁棒性能好的分类技术。

SILQ算法（改进C4.5算法）

（1）记录是预先排序的

（2）**SILQ是广度优先搜索算法**，而C4.5是深度优先。

预排序

- 对于连续属性在每个内部结点寻找其最优分裂标准时，都需要对训练集按照该属性的取值进行排序，操作需要大量时间。为此，SLIQ算法 采用了预排序技术，针对每个属性的取值，把所有的记录按照从小到大的顺序进行排序，以消除在决策树的每个结点对数据集进行的排序。

广度优先策略

- C4.5算法中，树的构造是按照深度优先策略完成的，需要对每个属性列表在每个结点处都进行一遍扫描，费时很多。SLIQ采用 广度优先策略构造决策树，即在决策树的每一层只需对每个属性列表扫描一次，就可以为当前决策树中每个叶子结点找到最优分裂标准。

分布式平台技术与分类算法结合

- **分布式改进算法：**
  - 局部挖掘
    - 在每个局部节点依据数据模型来收集当前数据， 然后对上一挖掘点所维 护的局部模型进行**增量式更新**，形成当前新的模式。
  - 模式传输
    - 当一个局部节点的模式更新完成后就通过网络把它传送到中心节点。
  - 全局挖掘
    - 当所有局部节点的当前模式都被成功地送到中心节点后，中心节点就进行**全局集成分类器**的学习，将全局模式更新到当前状态。

Spark随机森林

优化策略：切分点抽样、feature装箱、分区统计、逐层计算

Spark在实现随机森林时，采用了下面几个优化策略：

使用这些策略，原因在于RDD的数据分布在不同服务器上，为了避免过多的I/O，必须在原始算法上做出一些优化，否则执行时间可能难以接受。

流数据处理模型

界标模型、滑动窗口模型、快照模型

分治策略

- 数据分治与并行处理策略是大数据处理的基本策略。
- 但目前的分治与并行处理策略较少利用大数据的分布知识，且需考虑影响大数据处理的负载均衡与计算效率。
- 如何学习大数据的分布知识用于优化负载均衡是一个亟待解决的问题。

应用领域

- 目标客户细分
- 流失预警
- 征信评估
- 广告点击行为预测
- 垃圾邮件/短信处理
- 自动文本分类
- 安全领域入侵检测
- 自动驾驶
- ……