数据挖掘
第八讲· 分类

分类

1. 基本概念
2. 决策树归纳
3. 贝叶斯分类方法
4. 逻辑回归
5. 模型评估与选择
6. 提高分类准确率的技术

分类器准确率度量（1/4）

- 分类器M在给定检验集上的**准确率，acc(M)**：是分类器正确分类的检验集元组所占的百分比
  - 分类器M的误差率（误分类率）= 1 – acc(M)
  - 给定m个类，混淆矩阵（confusion matrix）中$CM_{i,j}$表示类i用分类器分到类j的元组数
  - 大部分元组应当用混淆矩阵对角线上的表目表示，非对角线上的表目接近于零
  - 有一些附加的行或列，提供每个类的合计或准确率

Actual class\Predicted class	C_1	¬ C_1
(Postive)C_1	**True Positives (TP)**	**False Negatives (FN)**
(Negtive)¬ C_1	**False Positives (FP)**	**True Negatives (TN)**

分类器准确率度量（2/4）

classes	buy_computer = yes	buy_computer = no	total	recognition (%)
buy_computer = yes	6954	46	7000	99.34
buy_computer = no	412	2588	3000	86.27
total	7366	2634	10000	95.42

分类器准确率度量（3/4）

- 几个术语
  - 正元组（感兴趣的主类元组，如buy_computer = yes）
  - 负元组（如 buy_computer = no）
  - 真正（true positive，TP）：分类器正确标记的正元组
  - 真负（true negative, TN）：分类器正确标记的负元组
  - 假正（false positive, FP）：错误标记的负元组
    - 如：buy_computer = no 的元组，分类器预测为buy_computer = yes
  - 假负（false negative,FN）：错误标记的正元组
    - 如：buy_computer = yes的元组，分类器预测为buy_computer = no

分类器准确率度量（4/4）

- 灵敏性（sensitivity）：正确识别的正元组的百分比 sensitivity = TP/pos
- 特效性（specificity）：正确识别的负元组的百分比 specificity = TN/neg
- 精度（precision） $precision = \frac{TP}{TP + FP}$
  - 如标记为“cancer”，实际是“cancer”的百分比

accuracy = sensitivity * pos/(pos + neg) + specificity * neg/(pos + neg)

❖ 准确率不合适的情况

- 如元组可能属于多个类时，准确率没有考虑这种情况

分类器准确率度量：补充

- **召回率Recall:** completeness – 正样本被预测为正的比例

$$recall = \frac{TP}{TP + FN}$$

- ***F* 度量 (*F_1* or *F*-score)**: 精度和召回率的调和均值

$$F = \frac{2 \times precision \times recall}{precision + recall}$$

- ***F_β*:** 精度和召回率的加权调和
  - ß 是非负实数，赋予召回率的权重是赋予精度权重的 ß 倍

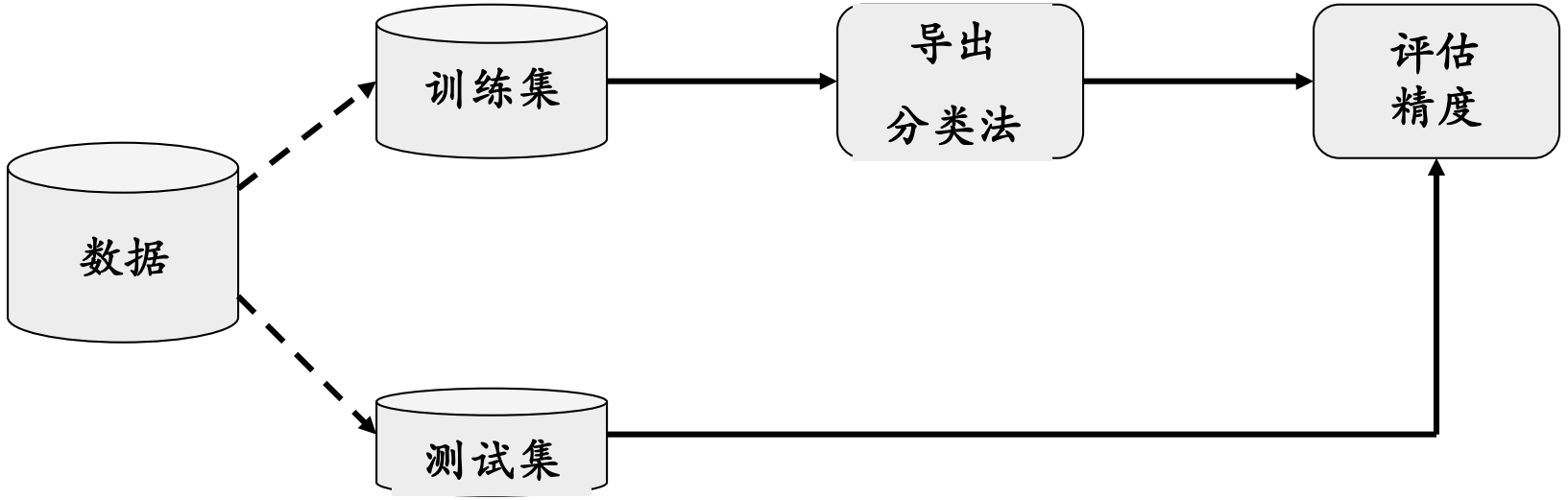
$$F_\beta = \frac{(1 + \beta^2) \times precision \times recall}{\beta^2 \times precision + recall}$$

比较分类器的其他方面

- 速度
  - 创建模型的速度和使用模型的速度
- 鲁棒性
  - 处理噪声和空缺值的能力
- 可伸缩性
  - 对大量数据的处理能力
- 可解释性
  - 模型的可理解程度

评估分类器或预测器的准确率（1/3）

- **保持方法（holdout）**
  - 给定数据随机地划分成两个独立的集合
    - 训练集：如2/3的数据，用于导出模型
    - 检验集：如1/3的数据，用于估计准确率



用保持方法估计准确率

评估分类器或预测器的准确率（2/3）

- **随机子抽样（random subsampling）：** 保持方法的一种变形
  - 将保持方法重复k次，总准确率估计取每次迭代准确率的平均值
- **交叉验证（Cross-validation，k折交叉验证中，k=10最常用）**
  - 初始数据随机划分成k个互不相交的子集，每个子集大小大致相等
  - 在第i次迭代，划分Di用作检验集，其余的划分一起用来训练模型，得到一个模型的准确率。总的准确率等于每次迭代的准确率的平均
- **留一（leave one out）：** 每次只给检验集留出一个样本
- **分层交叉确认（Stratified cross-validation）：** 折被分层，使得每个折中元组的类分布与在初始数据中的大致相同

评估分类器或预测器的准确率（3/3）

- **自助法（bootstrap method）**
  - 从给定训练元组中有放回地均匀抽样。对于小数据集，效果很好
- 常用的一种：.632自助法
  - 设给定数据集包含d个元组。有放回地抽样d次，产生d个样本的自助样本集或训练集。没有进入该训练集的元组最终形成检验集。
  - 平均情况下，63.2%的原数据元组将出现在自助样本中，而其余36.8%的元组将形成检验集、63.2%从何而来？ $(1 - 1/d)^d \approx e^{-1} = 0.368$
  - 重复抽样过程k次，模型的总体准确率为

$$acc(M) = \sum_{i=1}^{k} (0.632 \times acc(M_i)_{test\_set} + 0.368 \times acc(M_i)_{train\_set})$$

ROC曲线（1/4）

如何可视化两个分类模型的性能比较?

受试者工作特征曲线
（ROC曲线，
Receiver Operating Characteristics，
感受性曲线sensitivity curve）

- 起源于信号检测理论，第二次世界大战期间为雷达图像分析开发
- 显示给定模型的**真正率**（TPR，正确识别的正元组的比例）和**假正率**（FPR，不正确地识别为正元组的负元组的比例）之间的比较评定

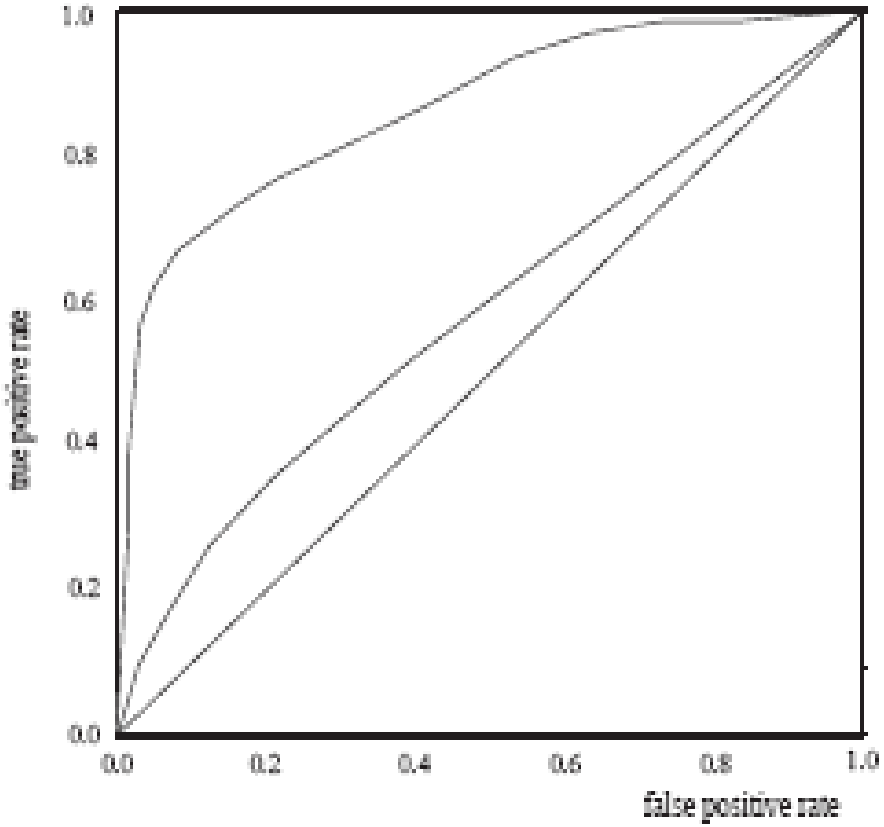
ROC曲线（2/4）

- **ROC曲线图反映的关系**
  - 横坐标X轴：FPR，假正率，X轴越接近零准确率越高；
  - 纵坐标Y轴：TPR，真正率（灵敏度），Y轴越大代表准确率越好
- **曲线下方部分的面积被称为AUC（Area Under Curve）**
  - 表示预测准确性
  - AUC值越高，说明预测准确率越高
  - 曲线越接近左上角（X越小，Y越大），预测准确率越高

ROC曲线（3/4）

- 对检验元组按递减序排序：分类器认为最可能属于正类（“yes”类）的元组出现在列表顶部
  - 垂直轴表示真正率
  - 水平轴表示假正率
  - 同时给出了对角线

✓ 从零开始，绘制ROC曲线；
✓ 若真正元组，TP增加，向上走一步，画一个点；
✓ 若负元组分为正，即假正，向右画一个点；

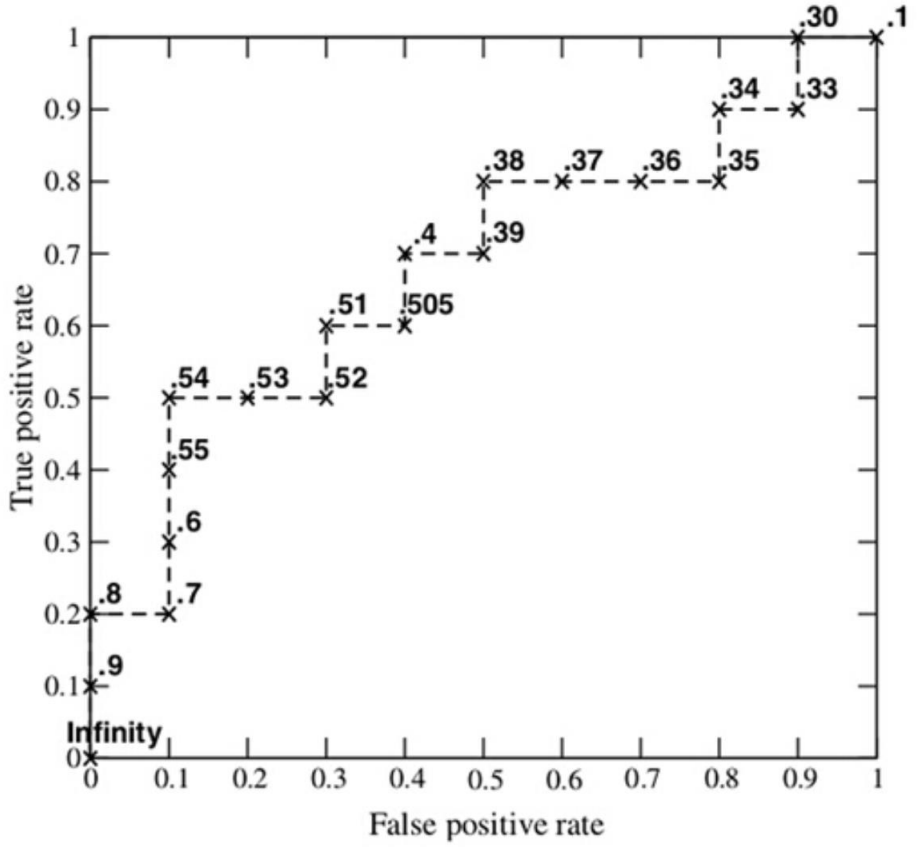


✓ 模型的ROC曲线离对角线越近，模型的准确率越低
✓ ROC曲线下方面积是模型准确率的度量
✓ 完全准确的模型的面积为1

ROC曲线（4/4）

举例

Inst#	Class	Score	Inst#	Class	Score
1	p	.9	11	p	.4
2	p	.8	12	n	.39
3	n	.7	13	p	.38
4	p	.6	14	n	.37
5	p	.55	15	n	.36
6	p	.54	16	n	.35
7	n	.53	17	p	.34
8	n	.52	18	n	.33
9	p	.51	19	p	.30
10	n	.505	20	n	.1



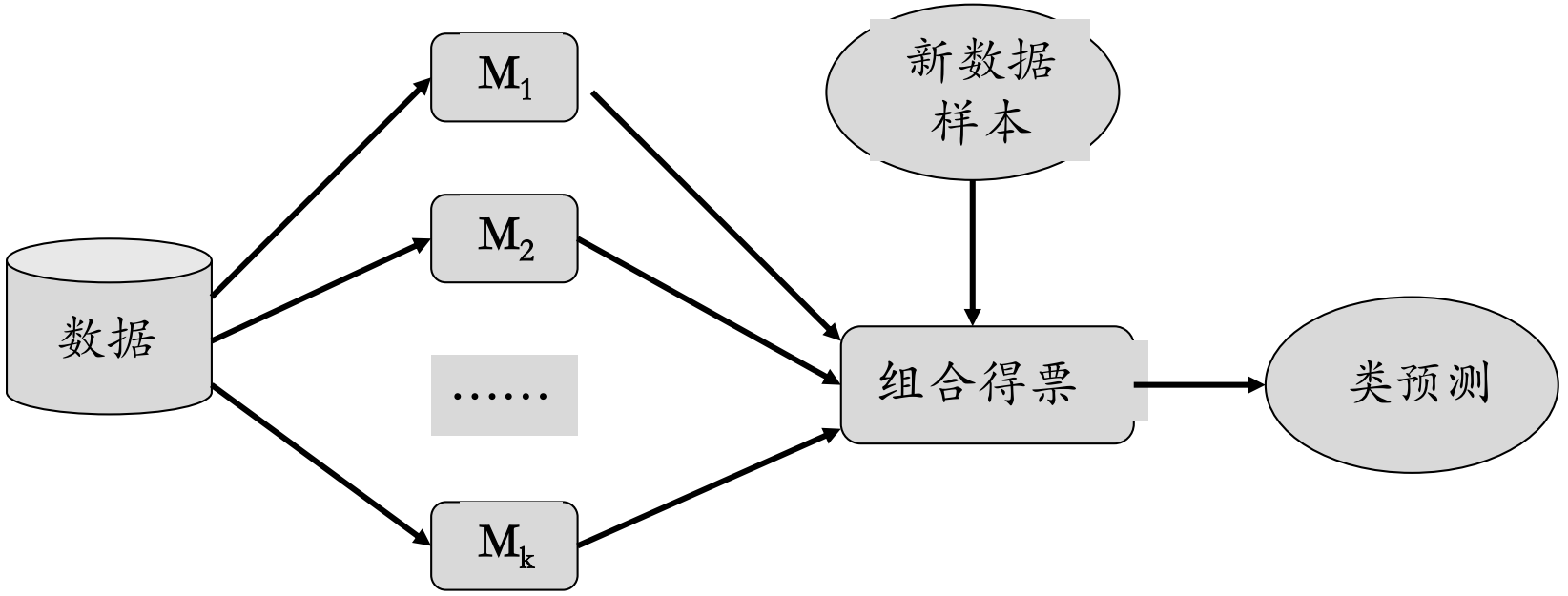
https://www.cnblogs.com/dlml/p/4403482.html

分类

1. 基本概念
2. 决策树归纳
3. 贝叶斯分类方法
4. 逻辑回归
5. 模型评估与选择
6. 提高分类准确率的技术

组合方法(Ensemble Methods)—提高准确率

- 使用模型的组合来提高准确率
- 将k个学习得到的模型（分类器或预测器）系列$M_1,M_2\cdots,M_K$组合起来，创建一个改进的复合模型M^*



组合方法（Ensemble Method）

- 不是一种单独的机器学习算法，而是**通过构建并结合多个机器学习器**来完成学习任务。
- 集百家之所长，拥有较高的准确率
- **不足之处：**模型的训练过程可能比较复杂，效率不高。
- **常见的组合方法：**
  - 基于Bagging的算法： bootstrap aggregating的缩写， 代表算法有随机森林;
  - 基于Boosting的算法：代表算法则有Adaboost、GBDT、XGBOOST等

装袋

提升

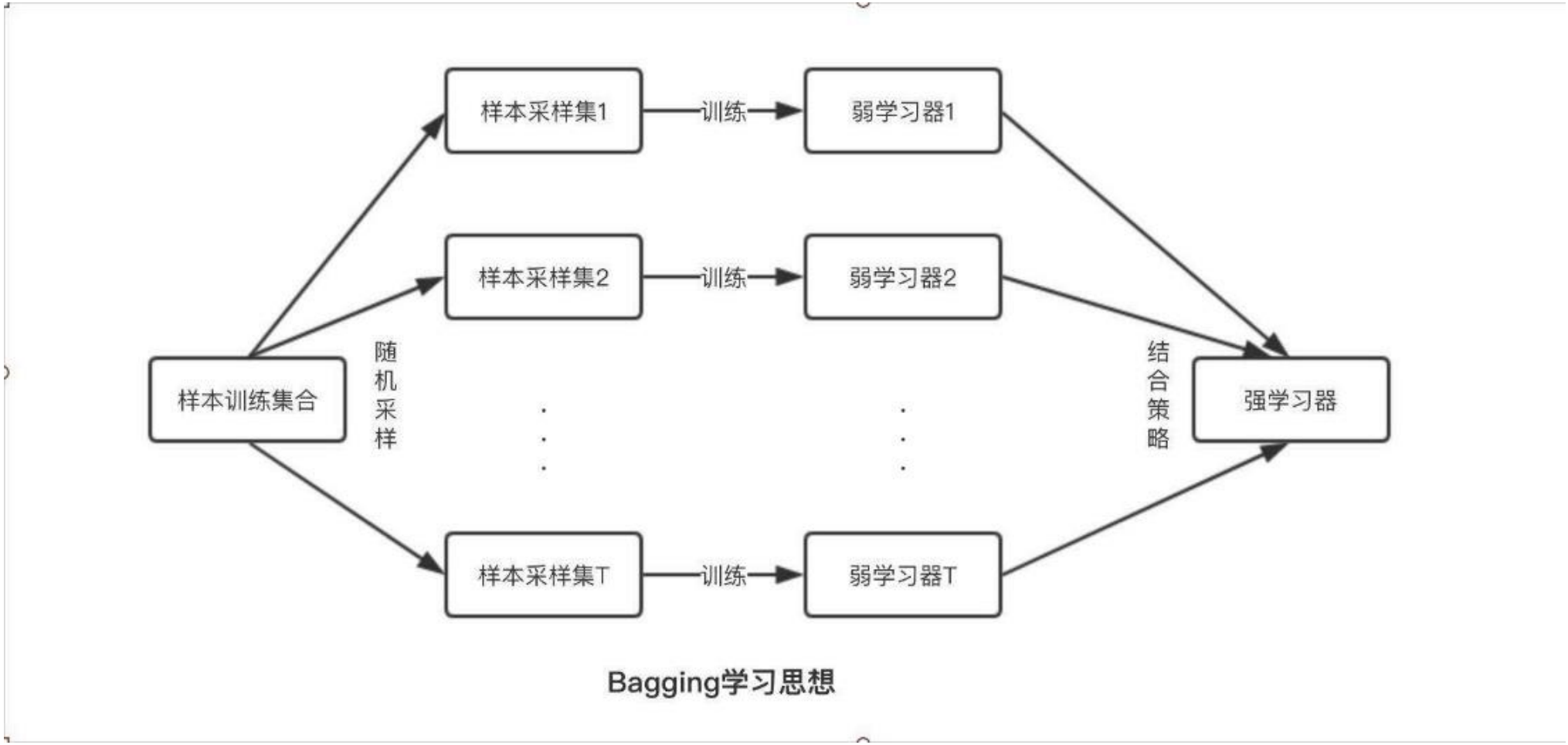
装袋（Bagging 1/3）

多数医生的表决比少数医生的表决更可靠

- 训练
  - 给定d个元组的集合D，对于迭代i，有放回地抽样d个元组的训练集D_i
  - 由每个训练集D_i学习，得到一个分类模型M_i

- 分类：对一个未知元组X分类
  - 每一个分类器M_i返回它的类预测，算作一票
  - 装袋分类器M^*统计得票，并将得票最高的类赋予X

装袋（Bagging 2/3）



Bagging学习思想

装袋（Bagging 3/3）

- **预测**
  - 通过取给定检验元组的每个预测的平均值，装袋也可以用于连续值的预测
- **准确率**
  - 通常比从原训练数据集D导出的单个分类器的准确率显著地高；
  - 对噪声数据的影响，它不会很差且鲁棒性较好。

提升（Boosting）

- **类比：**根据医生先前的诊断准确率，对每位医生的诊断赋予一个权重，加权诊断的组合作为最终的诊断
- **如何工作?**
  - 赋予每个训练元组一个权重
  - 迭代地学习k个分类器序列
  - 学习得到分类器M_i后，更新权重，使得其后的分类器M_{i+1}“**更关注**”M_i误分类的训练元组
  - 最终提升的分类器M^***组合**每个个体分类器，其中每个分类器**投票的权重是其准确率的函数**
- 可以扩充提升算法，预测连续值
- 与装袋方法相比，提升法能获得更高的准确率，但有可能过分拟合误分类的数据

提升算法(Boosting)

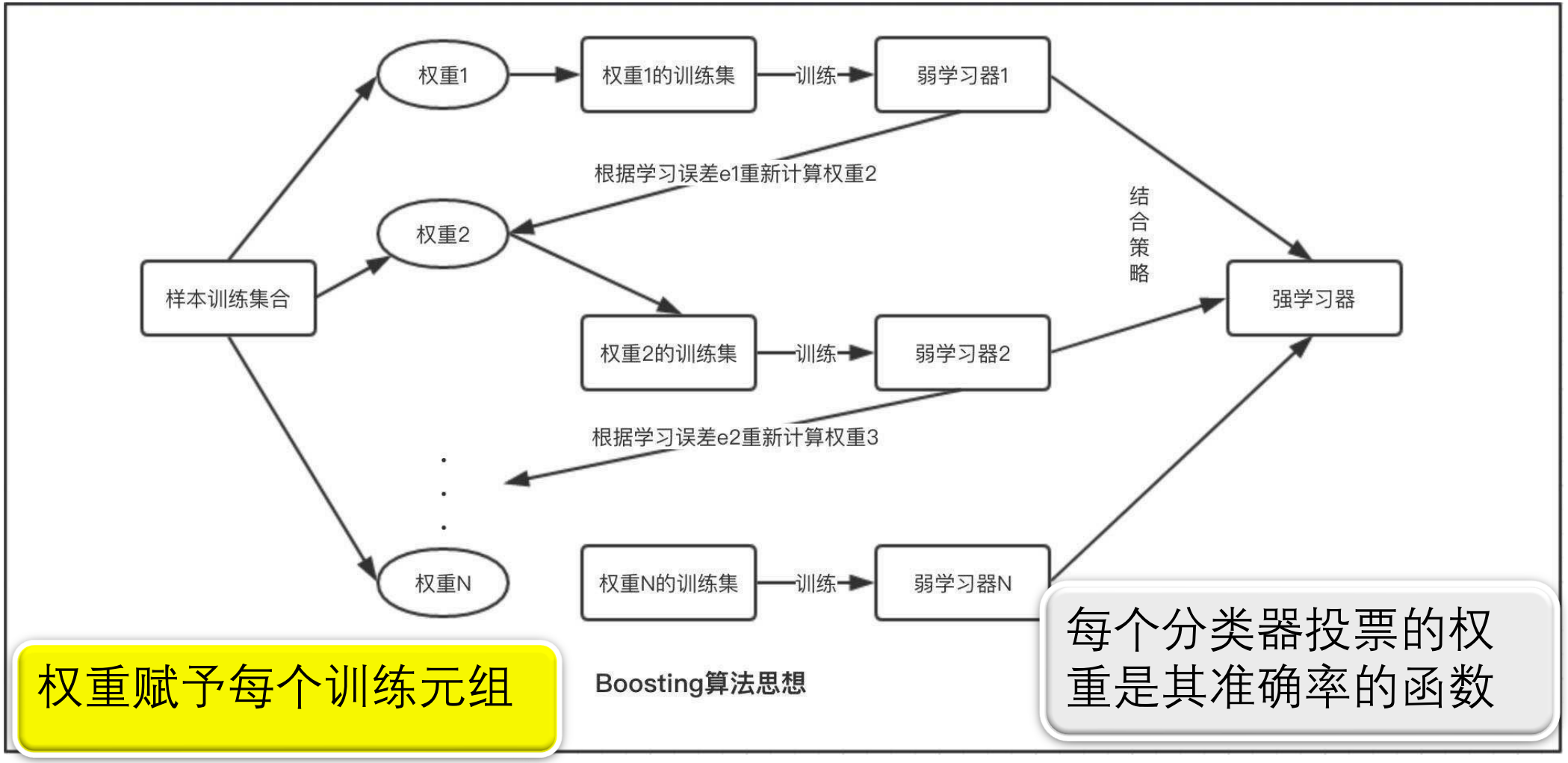
- 是常用的有效的统计学习算法，属于迭代算法
- 不断使用一个弱学习器，弥补前一个弱学习器的“不足”，来**串行地**构造一个较强的学习器，达到 使目标函数值 足够小

基本思想

1.先赋予每个训练样本相同的概率；
2.然后进行T次迭代，每次迭代后，对分类错误的样本加大权重(重采样)，使得在下一次的迭代中更加关注这些样本。

- Boosting系列算法
  - 最著名算法主要有**AdaBoost算法**
  - **提升树(boosting tree)系列算法**：应用最广泛的是梯度提升树(Gradient Boosting Tree)

提升算法(Boosting) 示意图



权重赋予每个训练元组

每个分类器投票的权重是其准确率的函数

Adaboost提升算法（1/2）

- Adaptive Boosting: Boosting + 单层决策树
- **基本思想：**通过训练数据的分布构造一个分类器，然后通过误差率求出这个若弱分类器的权重，通过更新训练数据的分布，迭代进行，直到达到迭代次数或者损失函数小于某一阈值。

- **步骤：**
  - 给定数据集D，包含d个类标记元组($\mathbf{X_1}$, y_1), …, ($\mathbf{X_d}$, y_d)
  - 初始对每个训练元组赋予相等的权重1/d
  - 产生集成学习的k个分类器需要执行k轮，
    - 第i轮从D中元组抽样，形成大小为d的训练集D_i
    - 每个元组被选中的机会由它的权重决定
    - 从训练元组D_i导出分类器模型M_i
    - 使用D_i作为检验集计算M_i的误差
    - **如果元组分类错误，则其权重增加，反之减少**

Adaboost提升算法（2/2）

- **误差率（error rate）：** 模型M_i的误差率是M_i误分类的D_i中所有元组的加权和，即

$$error(M_i) = \sum_{j}^{d} w_j \times err(\mathbf{X_j})$$

  - err(X_j)是元组X_j的误分类误差：如果误分类，则为1，否则为0
- **分类器Mi的表决权重为**

$$\log \frac{1-error(M_i)}{error(M_i)}$$

  - 对于每个类c，对每个将类c指派到X的分类器的权重求和。
  - 具有和最大的类是“赢家”，并返回作为元组X的预测值

集成学习之结合策略

- **平均法**
  - 对于数值类的回归预测问题，通常使用平均法策略/加权平均法，即：对于若干弱学习器的输出进行平均得到最终的预测输出。
- **投票法**
  - 相对多数投票法
  - 绝对多数投票法（过半数）
  - 加权投票法

集成学习之结合策略

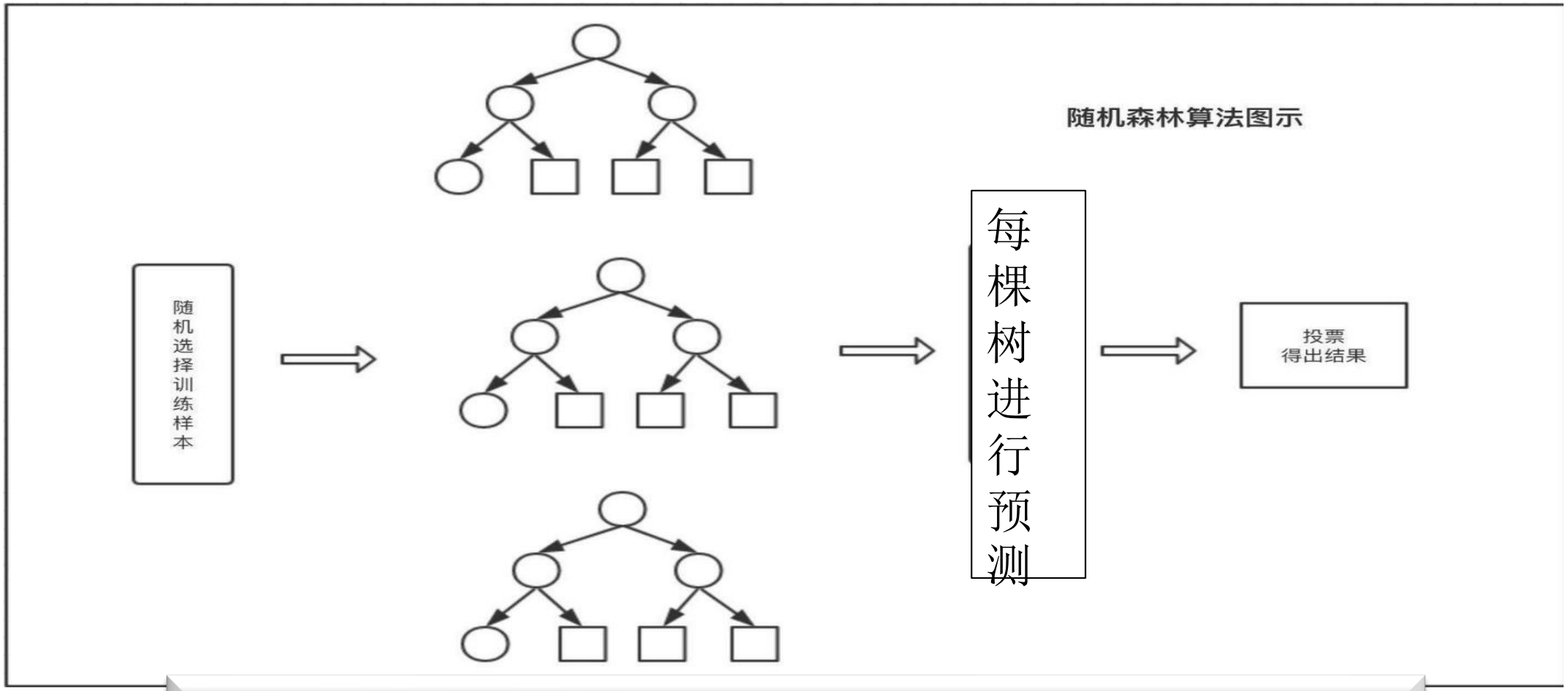
- **学习法：**代表方法是stacking：一种有层次的融合模型
  - 不是对弱学习器的结果做简单的逻辑处理，而是再加上一层学习器，也即将训练集弱学习器的学习结果作为输入，将训练集的输出作为输出，重新训练一个学习器来得到最终结果。
    - 弱学习器称为初级学习器，用于结合的学习器称为次级学习器（通常为线性模型LR）
    - 对于测试集，首先用初级学习器预测一次，得到次级学习器的输入样本，再用次级学习器预测一次，得到最终的预测结果。
  - **不同组合结果：**
    - Bagging + 决策树 = 随机森林
    - AdaBoost + 决策树 = 提升树
    - Gradient Boosting + 决策树 = GBDT，

    其中GBDT在达观数据个性化推荐重排序层得到很好地应用

随机森林

- 每个分类器都是一个决策树。个体决策树在每个结点使用随机选择的属性来确定划分。每棵树都投票，并返回得票最多的类。
- 构建随机森林的两种方法：
  - **Forest-RI（随机输入选择）**： 在每个结点上随机选择F个属性作为节点处分割的候选。CART方法用于将树生长到最大。
  - **Forest-RC（随机线性组合）**： 创建新属性（或特征）， 这些属性是现有属性的线性组合（减少单个分类器之间的相关性）
- **优势：**
  - 精度可与Adaboost媲美，但对误差和异常值更具鲁棒性
- 思考：随机森林算法对每次拆分时选择的属性数量 敏感吗?

随机森林示意图



- 构建决策树的时候分裂节点的选择是依据最小基尼系数
- 在构建决策树的过程中不需要剪枝
- 整个森林树的数量和每棵树的特征需要人为进行设定

小结

分类

评估分类和预测方法的五条标准

- ✓ 准确率
- ✓ 计算速度
- ✓ 鲁棒性
- ✓ 可伸缩性
- ✓ 可解释性

决策树算法

ID3、C4.5、CART

朴素贝叶斯分类

基于后验概率的贝叶斯定理

逻辑回归分类

评估分类准确率的方法

- ✓ 推荐方法：分层的k-折交叉确认
- ✓ 提高整体准确率方法：装袋和提升
- ✓ 准确率度量的替换：灵敏性、特效性和精度