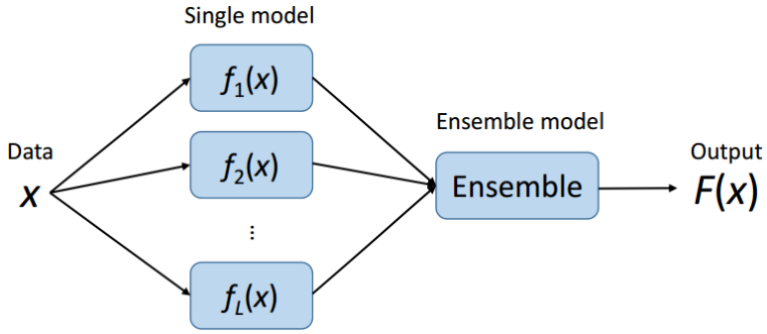
# 集成学习（ensemble learning)

集成学习，其英文名称叫做(**ensemble learning**)，它通过将多个学习器集成在一起来达到学习的目的。主要是将有限的模型相互组合，其名称有时也会有不同的叫法，有时也会被称为多分类器系统(**multi-classifier system**)、委员会学习(**committee learning**)、Modular systems、classifier fusion、combination、aggregation等。这些概念相互之间互相联系，又有些许区别，对于概念的定义业界还没有达成共识。整个算法所表现出来的性能非常地强悍，许多高水平的竞赛(Knowledge Discovery and Data Mining、Kaggle)中都是首选。

在机器学习，满足训练集的假设不一定在实际应用中有同样好的表现，这样学习算法选择哪个假设进行输出的时候就面临着一定的风险，把多个假设集成起来能够降低这种风险（这可以理解为通过集成使得各个假设和目标假设之间的误差得到一定程度的抵消）。




在周志华西瓜书中通过Hoeffding不等式证明了，**随着集成中个体分类器数目的增大，集成的错误率将指数级下降，最终趋于零。**

集成学习先产生一组“个体学习器”(**individual learner**)，再通过某种策略将其结合起来。依据每个个体学习器所采用的学习算法是否相同，可以分为**同质集成**和**异质集成。**

这里集成一般有两种：**同质和异质。**同质是指个体学习器全是同一类型，这种同质集成中的个体学习器又称“基学习器”。异质是指个体学习器包含不同类型得学习算法，比如同时包含决策树和神经网络。一般我们常用的都是同质的，即个体学习器都是同一类型的。

- 同质集成中，个体学习器由**相同**的学习算法生成，个体学习器称为**基学习器。**
- 异质集成中，个体学习器由**不同**的学习算法生成，个体学习器称为**组件学习器。**

**好而不同**

集成学习器性能要好于单个个体学习器需要满足**好而不同**的两点要求：

1. 个体学习器要好于随机猜测的结果。
2. 个体学习器要相互独立。

第一个条件相对来说比较容易实现，在当前问题下训练一个模型，结果比瞎猜的结果好就行了。**第二个条件是集成学习研究的核心问题。**每个个体学习器学习的都是同一个问题，所以个体学习器不可能做到完全相互独立.

预测方法：平均，权重平均，线性回归分类，乘积，多层次，树模型

# 集成学习常用方法

**首先考虑输入。**如果每个学习器学习不同的样本，那么可以学习出相对来说不同的个体学习器。那么现在的问题就是怎么划分训练样本，你可以随机抽取，或者利用不同的属性子集训练出不同的个体学习器。

**其次考虑模型**，如果基学习器的模型不一样，也能训练出不同的个体学习器。

**最后考虑输出**，如果我们依据标签的特性来进行划分，也能得到不同的个体学习器。

并行集成方法：（其中参与训练的基础学习器并行生成（例如 Random Forest）。并行方法的原理是利用基础学习器之间的独立性，通过平均可以显著降低错误。）

- bagging（自然聚合）（套袋法）
- Random Forest（随机森林）

序列集成方法：（其中参与训练的基础学习器按照顺序生成（例如 AdaBoost）。序列方法的原理是利用基础学习器之间的依赖关系。通过对之前训练中错误标记的样本赋值较高的权重，可以提高整体的预测效果）

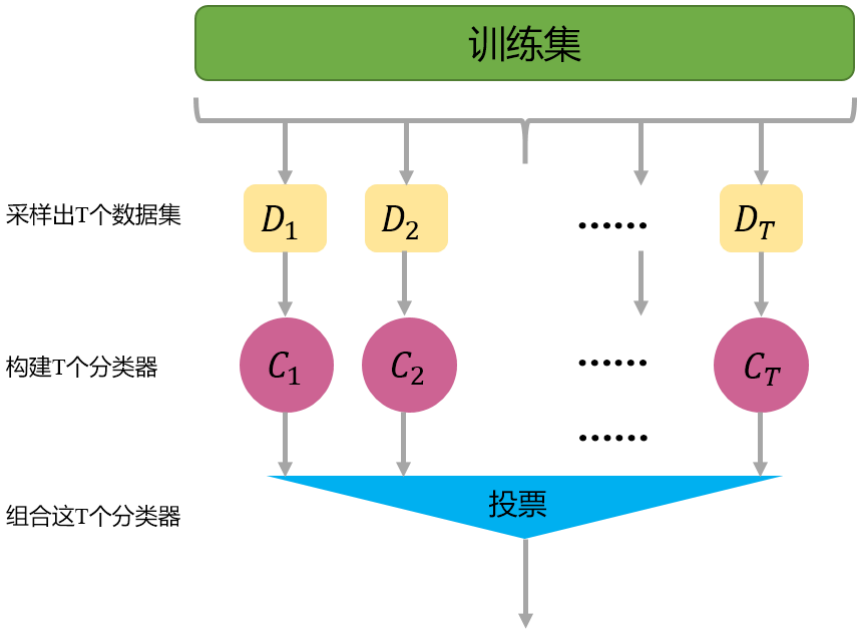
- Boosting (Adaboost)（提升法）
- Gradient Boosting Decision Tree

# Bagging（boostrap aggregating)

通过组合多个模型来减少泛化误差

bootstrap也称为自助法，它是一种有放回的抽样方法，目的为了得到统计量的分布以及置信区间

Bagging的主要思想如下图所示，首先从数据集中采样出T个数据集，然后基于这T个数据集，每个训练出一个基分类器，再讲这些基分类器进行组合做出预测。Bagging在做预测时，对于分类任务，使用简单的投票法。对于回归任务使用简单平均法。若分类预测时出现两个类票数一样时，则随机选择一个。




**总结一下bagging方法：**

① Bagging通过降低基分类器的方差，改善了泛化误差

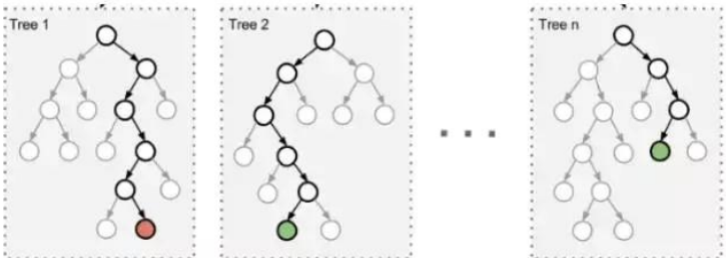
② 其性能依赖于基分类器的稳定性；如果基分类器不稳定，bagging有助于降低训练数据的随机波动导致的误差；如果稳定，则集成分类器的误差主要由基分类器的偏倚引起

③ 由于每个样本被选中的概率相同，因此bagging并不侧重于训练数据集中的任何特定实例

# Random Forest

在随机森林中，集成中的每棵树都是由从训练集中抽取的样本（即 bootstrap 样本）构建的。另外，与使用所有特征不同，这里随机选择特征子集，从而进一步达到对树的随机化目的。

因此，随机森林产生的偏差略有增加，但是由于对相关性较小的树计算平均值，估计方差减小了，导致模型的整体效果更好。




CART分类回归树

优点：

1. 适用于回归和分类问题自然处理分类预测器计算简单，快速拟合，即使是大问题。
2. 可以处理高度非线性的相互作用和分类边界。
3. 如果树小，很容易解释。

缺点：

1. 准确性:当前的方法，如SVM和集成分类器的错误率通常比CART低30%。
2. 不稳定性:如果我们稍微改变数据，树形图就会发生很大变化。所以解释并不像看起来那么简单。

随机森林：

随机性的两个来源:

1. Bagging method :每个树都是用boostrap训练数据生长的，不同的自己
2. 随机向量法：在每个节点上，从m个属性的随机样本中选择最佳分割，而不是从所有属性中选择最佳分割。（每次分割的属性选择不同）

算法：

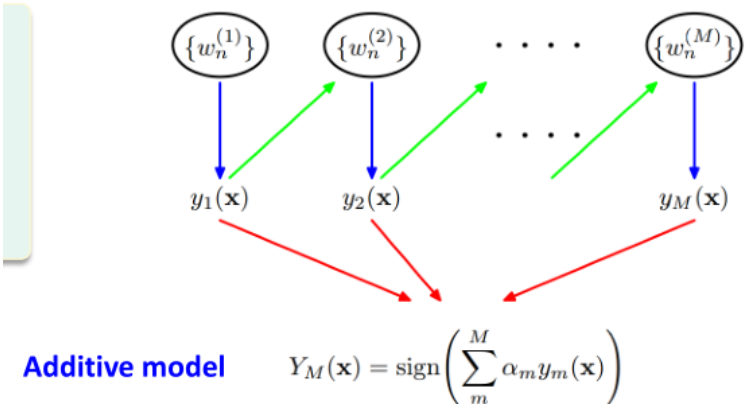
1. For b=1 to B:
   - Draw a bootstrap sample $\boldsymbol{D}^*$ of size $N$ from the training data
   - Grow a tree $T_b$ from $\boldsymbol{D}^*$, by recursively repeating the following steps
     For each terminal node of the tree, until the minimum node size $n_{min}$ is reached.
     - Select $m$ variables at random from the $d$ variables.
       - Typically $m = log_2 d$
     - Pick the best variable/split-point among the $m$.
     - Split the node into two daughter nodes
2. Output the ensemble of trees $\{T_b\}_1^B$.

To make a prediction at a new points,

- *Regression*: prediction average $\hat{f}_{rf}^B(x) = \frac{1}{B}\sum_{b=1}^B T_b(x)$
- *Classification*: majority voting $\hat{C}_{rf}^B(x) = \text{majority vote } \{\hat{C}_b(x)\}_1^B$.

# Boosting

其主要思想是将弱分类器组装成一个强分类器。在PAC（probably approximately correct，概率近似正确）学习框架下，则一定可以将弱分类器组装成一个强分类器。




关于Boosting的两个核心问题：

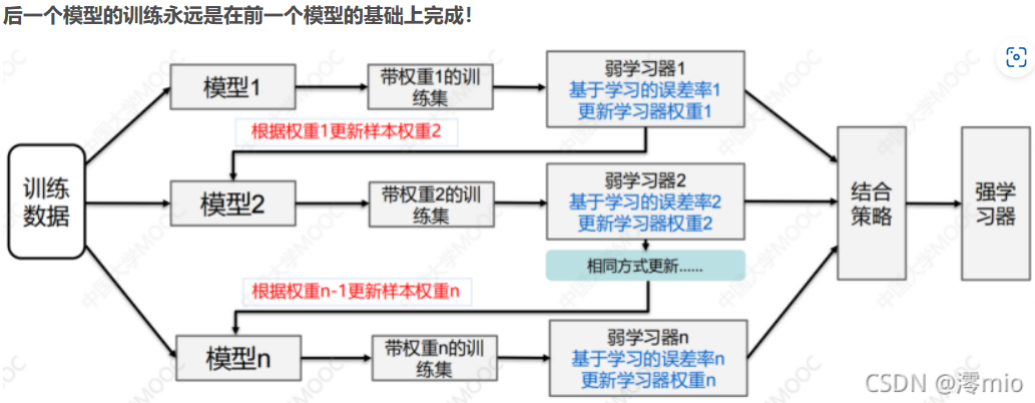
1）在每一轮如何改变训练数据的权值或概率分布？

通过提高那些在前一轮被弱分类器分错样例的权值，减小前一轮分对样例的权值，来使得分类器对误分的数据有较好的效果。

2）通过什么方式来组合弱分类器？

通过加法模型将弱分类器进行线性组合，比如：

**AdaBoost**（Adaptive boosting）算法：刚开始训练时对每一个训练例赋相等的权重，然后用该算法对训练集训练t轮，每次训练后，对训练失败的训练例赋以较大的权重，也就是让学习算法在每次学习以后更注意学错的样本，从而得到多个预测函数。通过拟合残差的方式逐步减小残差，将每一步生成的模型叠加得到最终模型。




具体内容在p155

adaboost

# Gradient Boosting Decision Tree（GBDT）