北风网
IBEIFENG.COM

人工智能之机器学习

集成学习：随机森林、GBDT

上海育创网络科技股份有限公司

主讲人：刘老师(GerryLiu)

课程要求

- 课上课下“九字”真言
  - 认真听，**善摘录，勤思考**
  - **多温故，乐实践**，再发散
- 四不原则
  - 不懒散惰性，不迟到早退
  - 不请假旷课，不拖延作业
- 一点注意事项
  - 违反“四不原则”，不推荐就业

课程内容

- 集成算法
- 随机森林
- 提升算法
- GBDT(迭代决策树)
- Adaboost
- Stacking

集成学习(Ensemble Learning)

- 集成学习的思想是将若干个学习器(分类器&回归器)组合之后产生一个新学习器。弱分类器(weak learner)指那些分类准确率只稍微好于随机猜测的分类器(error rate < 0.5)；
- 集成算法的成功在于保证弱分类器的多样性(Diversity)。而且集成不稳定的算法也能够得到一个比较明显的性能提升。
- 常见的集成学习思想有：
  - Bagging
  - Boosting
  - Stacking

集成学习(Ensemble Learning)

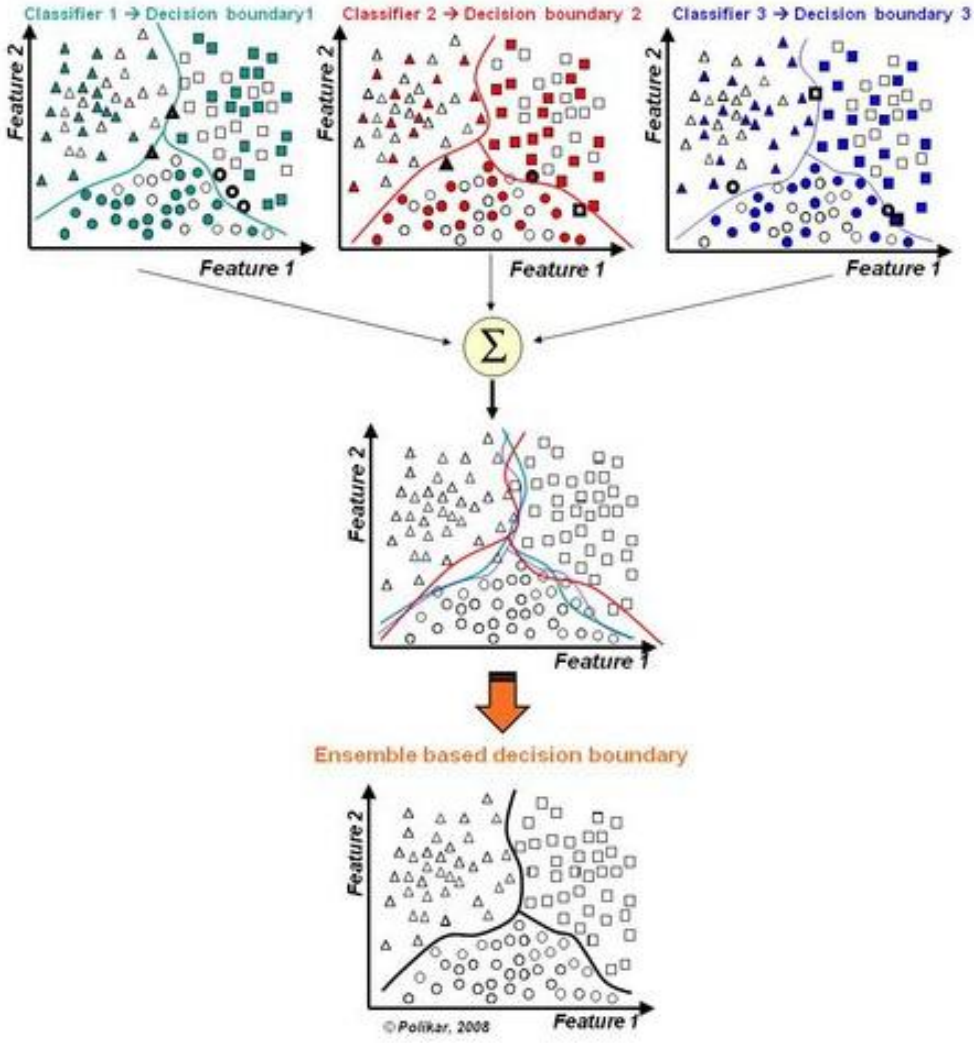


Figure 1: Combining an ensemble of classifiers for reducing classification error and/or model selection.

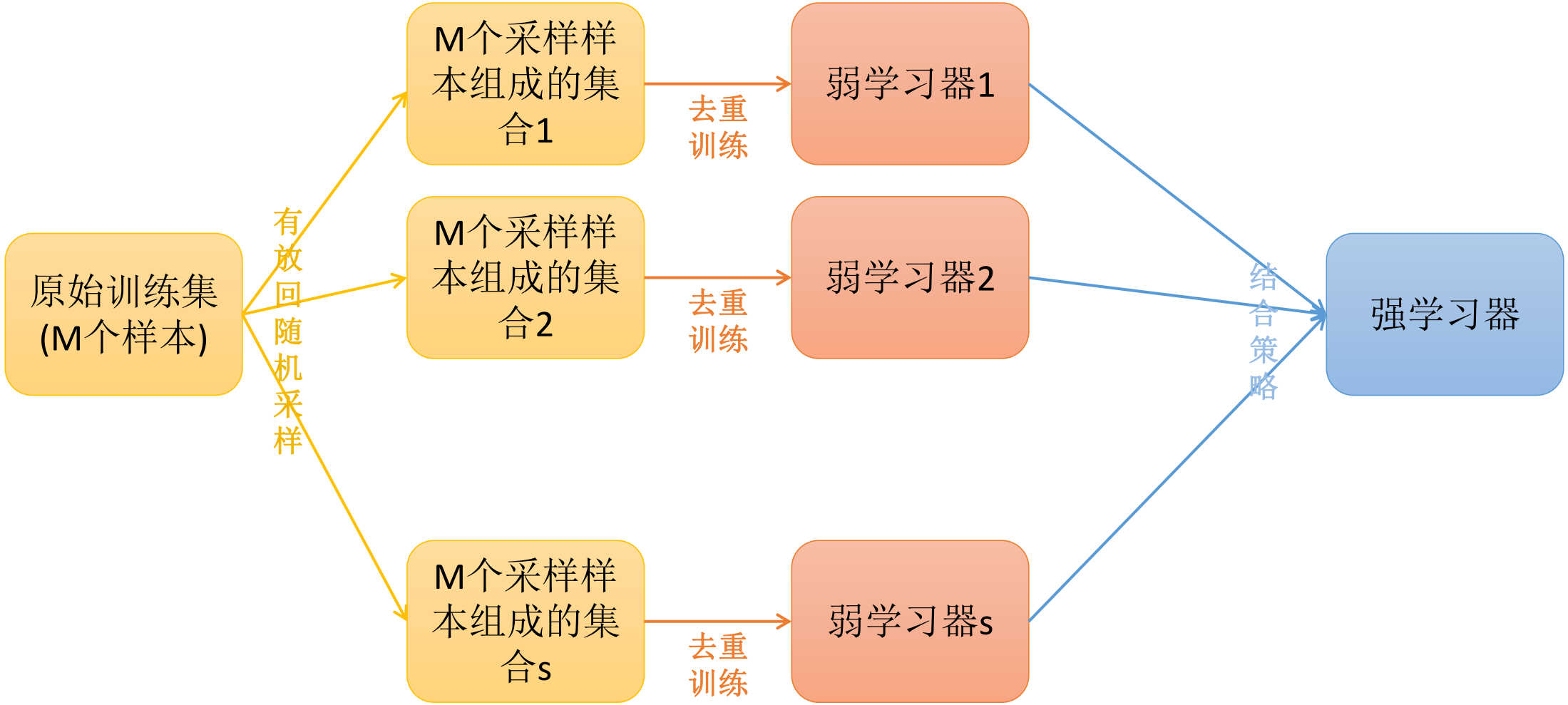
Why need Ensemble Learning?

- 1. 弱分类器间存在一定的差异性，这会导致分类的边界不同，也就是说可能存在错误。那么将多个弱分类器合并后，就可以得到更加合理的边界，减少整体的错误率，实现更好的效果；
- 2. 对于数据集过大或者过小，可以分别进行划分和有放回的操作产生不同的数据子集，然后使用数据子集训练不同的分类器，最终再合并成为一个大的分类器；
- 3. 如果数据的划分边界过于复杂，使用线性模型很难描述情况，那么可以训练多个模型，然后再进行模型的融合；
- 4. 对于多个异构的特征集的时候，很难进行融合，那么可以考虑每个数据集构建一个分类模型，然后将多个模型融合。

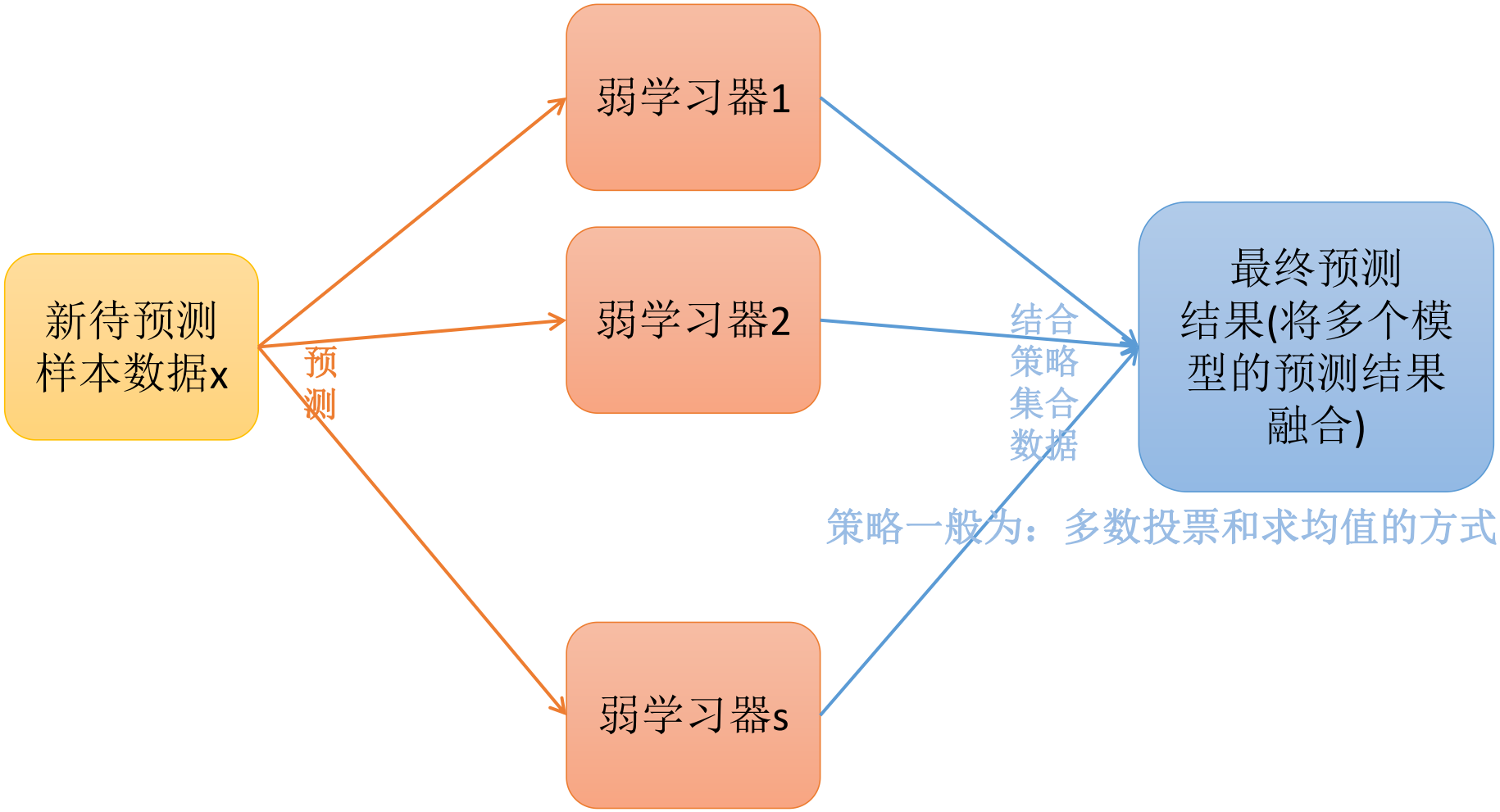
Bagging方法

- Bagging方法又叫做自举汇聚法(Bootstrap Aggregating)，思想是：在原始数据集上通过**有放回的抽样**的方式，重新选择出S个新数据集来分别训练S个分类器的集成技术。
- Bagging方法训练出来的模型在预测新样本分类/回归的时候，会使用**多数投票**或者**求均值**的方式来统计最终的分类/回归结果。
- Bagging方法的弱学习器可以是基本的算法模型，eg: Linear、Ridge、Lasso、Logistic、Softmax、ID3、C4.5、CART、SVM、KNN等。
- 备注：Bagging方式是有放回的抽样，并且每个子集的样本数量必须和原始样本数量一致，所以抽取出来的子集中是存在重复数据的，但是在模型训练的时候会将重复数据删除（相当于去重distinct），也就是说真正用于训练模型的数据集样本和原始样本数是不一致。

Bagging方法_训练过程



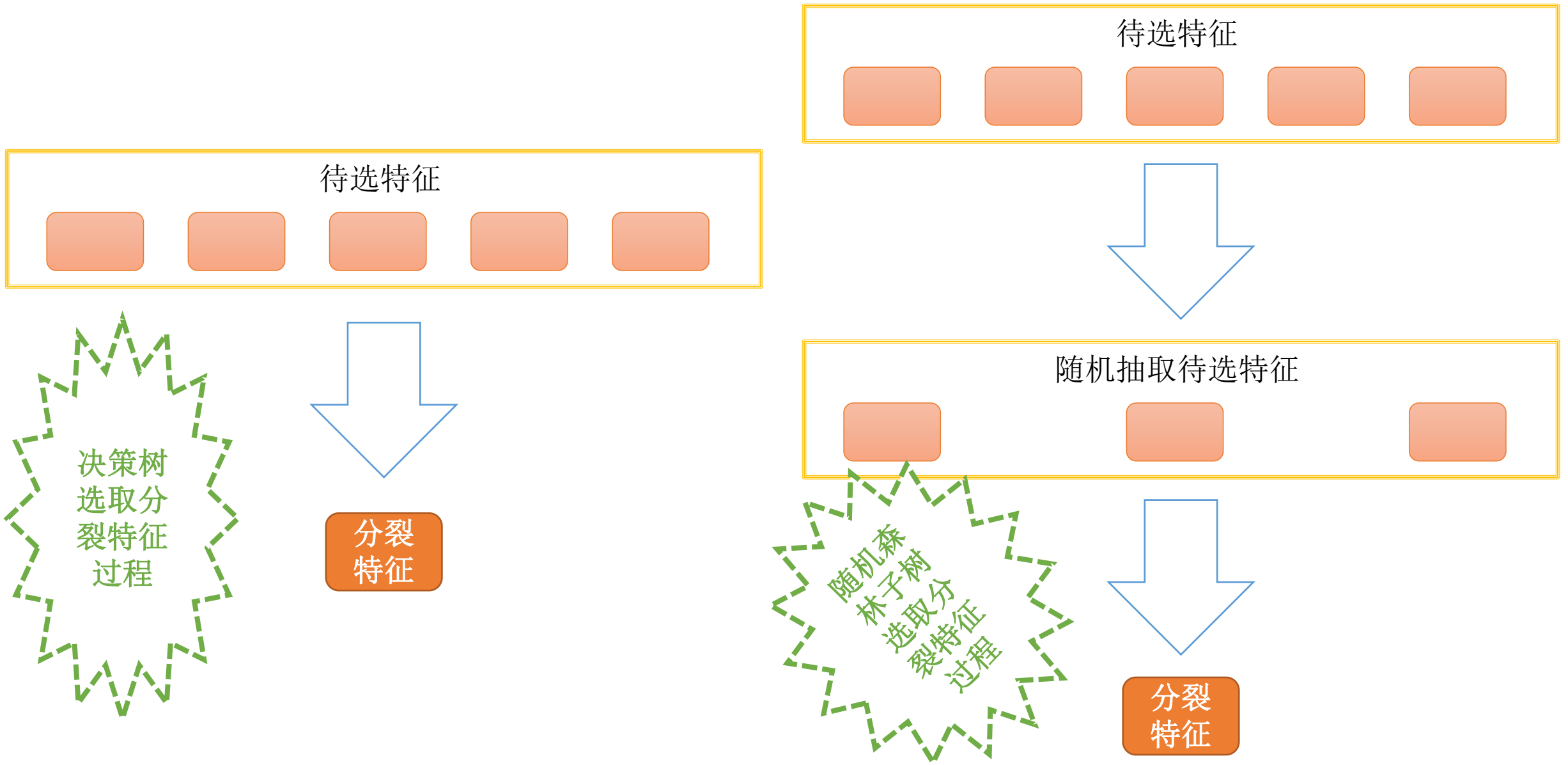
Bagging方法_预测过程



随机森林(Random Forest)

- 在Bagging策略的基础上进行修改后的一种算法
  - 1. 从原始样本集(n个样本)中用Bootstrap采样(有放回重采样)选出n个样本；真正用于模型训练的是这抽取出来的样本去重之后的数据集，也就是一般情况用户模型训练的样本数目实际上不等于n，应该是小于n。
  - 2. 使用抽取出来的子数据集(去重后的)来训练决策树；从所有属性中随机选择K个属性，从K个属性中选择出最佳分割属性作为节点来迭代的创建决策树
  - 3. 重复以上两步m次，即建立m棵决策树；
  - 4. 这m个决策树形成随机森林，通过投票表决结果决定数据属于那一类

随机森林(Random Forest)



RF的推广算法

- RF算法在实际应用中具有比较好的特性，应用也比较广泛，主要应用在：分类、回归、特征转换、异常点检测等。常见的RF变种算法如下：
  - Extra Tree
  - Totally Random Trees Embedding(TRTE)
  - Isolation Forest

Extra Tree

- Extra Tree是RF的一个变种，原理基本和RF一样，区别如下：
  - 1. RF会随机重采样来作为子决策树的训练集，而Extra Tree每个子决策树采用原始数据集训练；
  - 2. RF在选择划分特征点的时候会和传统决策树一样，会基于信息增益、信息增益率、基尼系数、均方差等原则来选择最优特征值；而Extra Tree会随机的选择一个特征值来划分决策树。
- Extra Tree因为是随机选择特征值的划分点，这样会导致决策树的规模一般大于RF所生成的决策树。也就是说Extra Tree模型的方差相对于RF进一步减少。在某些情况下，Extra Tree的泛化能力比RF的强。

Totally Random Trees Embedding(TRTE)

- TRTE是一种非监督的数据转化方式。将低维的数据集映射到高维，从而让映射到高维的数据更好的应用于分类回归模型。
- TRTE算法的转换过程类似RF+KDTree算法的方法，建立T个决策树来拟合数据(是类似KD-Tree一样基于特征属性的方差选择划分特征)。当决策树构建完成后，数据集里的每个数据在T个决策树中叶子节点的位置就定下来了，将位置信息转换为向量就完成了特征转换操作。
- 案例：有3棵决策树，各个决策树的叶子节点数目分别为:5,5,4，某个数据x划分到第一个决策树的第3个叶子节点，第二个决策树的第一个叶子节点，第三个决策树的第四个叶子节点，那么最终的x映射特征编码为:(0,0,1,0,0, 1,0,0,0,0, 0,0,0,1)

Isolation Forest(IForest)

- IForest是一种异常点检测算法，使用类似RF的方式来检测异常点；IForest算法和RF算法的区别在于：
  - 1. 在随机采样的过程中，一般只需要少量数据即可；
  - 2. 在进行决策树构建过程中，IForest算法会随机选择一个划分特征，并对划分特征随机选择一个划分阈值；
  - 3. IForest算法构建的决策树一般深度max_depth是比较小的。
- 区别原因：目的是异常点检测，所以只要能够区分异常的即可，不需要大量数据；另外在异常点检测的过程中，一般不需要太大规模的决策树。

Isolation Forest(IForest)

- 对于异常点的判断，则是将测试样本x拟合到m棵决策树上。计算在每棵树上该样本的叶子节点的深度$h_t(x)$。从而计算出平均深度h(x)；然后就可以使用下列公式计算样本点x的异常概率值，p(s,m)的取值范围为[0,1]，越接近于1，则是异常点的概率越大。备注：如果落在的叶子节点为正常样本点，那么当前决策树不考虑，如果所有决策树上都是正常样本点，那么直接认为异常点概率为0.

$$p(x,m)=2^{-\frac{h(x)}{c(m)}}$$

$c(m)=2\ln(m-1)+\xi-2\frac{m-1}{m}$；$m$为样本个数，$\xi$为欧拉常数

RF随机森林总结

- RF的主要优点：
  - 1. 训练可以并行化，对于大规模样本的训练具有速度的优势；
  - 2. 由于进行随机选择决策树划分特征列表，这样在样本维度比较高的时候，仍然具有比较高的训练性能；
  - 3. 给以给出各个特征的重要性列表；
  - 4. 由于存在随机抽样，训练出来的模型方差小，泛化能力强；
  - 5. RF实现简单；
  - 6. 对于部分特征的缺失不敏感。
- RF的主要缺点：
  - 1. 在某些噪音比较大的特征上（数据特别异常情况），RF模型容易陷入过拟合；
  - 2. 取值比较多的划分特征对RF的决策会产生更大的影响，从而有可能影响模型的效果。

随机森林算法案例

- 使用随机森林算法API对乳腺癌数据进行分类操作，根据特征属性预测是否会得乳腺癌的四个目标属性的值，并理解随机森林中决策树数量和决策树深度对模型的影响
- 数据来源：乳腺癌数据

(bool) Hinselmann: target variable
(bool) Schiller: target variable
(bool) Cytology: target variable
(bool) Biopsy: target variable

目标属性

Cervical cancer (Risk Factors) Data Set

Download: Data Folder, Data Set Description

Abstract: This dataset focuses on the prediction of indicators/diagnosis of cervical cancer. The features cover demographic information, habits, and historic medical records.

Data Set Characteristics:	Multivariate	**Number of Instances:**	858	**Area:**	Life
Attribute Characteristics:	Integer, Real	**Number of Attributes:**	36	**Date Donated**	2017-03-03
Associated Tasks:	Classification	**Missing Values?**	Yes	**Number of Web Hits:**	3687

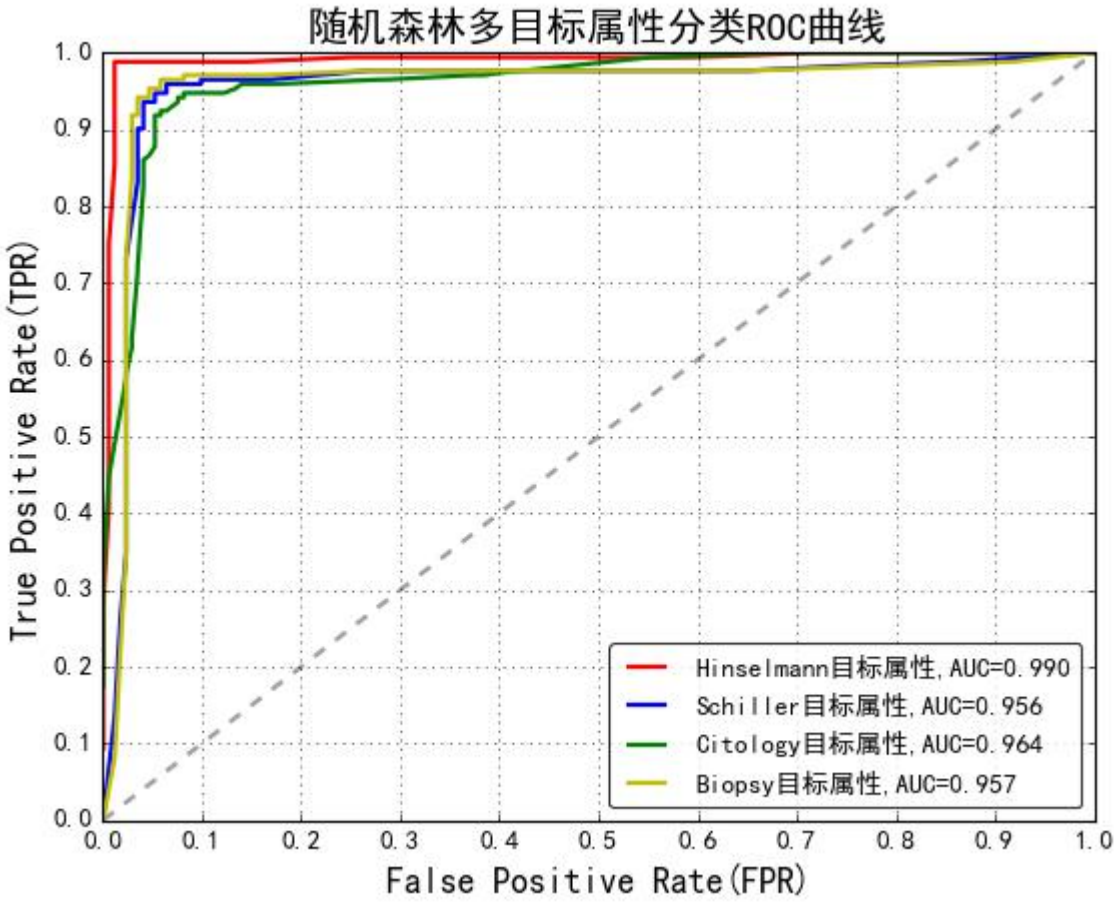
class sklearn.ensemble.**RandomForestClassifier**(*n_estimators=10*, *criterion='gini'*, *max_depth=None*, *min_samples_split=2*, *min_samples_leaf=1*, *min_weight_fraction_leaf=0.0*, *max_features='auto'*, *max_leaf_nodes=None*, *bootstrap=True*, *oob_score=False*, *n_jobs=1*, *random_state=None*, *verbose=0*, *warm_start=False*, *class_weight=None*) [source]

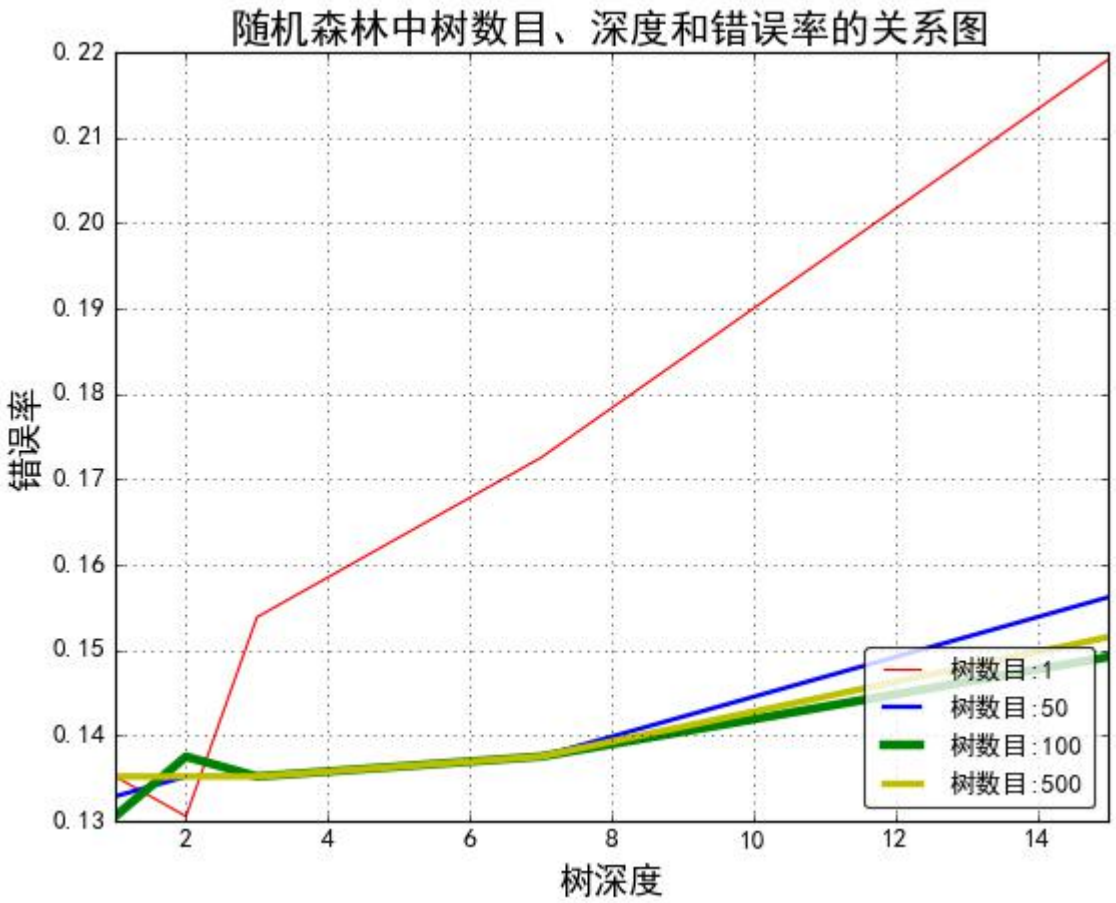
Attribute Information:

(int) Age
(int) Number of sexual partners
(int) First sexual intercourse (age)
(int) Num of pregnancies
(bool) Smokes
(bool) Smokes (years)
(bool) Smokes (packs/year)
(bool) Hormonal Contraceptives
(int) Hormonal Contraceptives (years)
(bool) IUD
(int) IUD (years)
(bool) STDs
(int) STDs (number)
(bool) STDs:condylomatosis
(bool) STDs:cervical condylomatosis
(bool) STDs:vaginal condylomatosis
(bool) STDs:vulvo-perineal condylomatosis
(bool) STDs:syphilis
(bool) STDs:pelvic inflammatory disease
(bool) STDs:genital herpes
(bool) STDs:molluscum contagiosum
(bool) STDs:AIDS
(bool) STDs:HIV
(bool) STDs:Hepatitis B
(bool) STDs:HPV
(int) STDs: Number of diagnosis
(int) STDs: Time since first diagnosis
(int) STDs: Time since last diagnosis
(bool) Dx:Cancer
(bool) Dx:CIN
(bool) Dx:HPV
(bool) Dx

特征属性

随机森林算法案例


随机森林多目标属性分类ROC曲线
True Positive Rate(TPR)
False Positive Rate(FPR)
Hinselmann目标属性, AUC=0.990
Schiller目标属性, AUC=0.956
Citology目标属性, AUC=0.964
Biopsy目标属性, AUC=0.957


随机森林中树数目、深度和错误率的关系图
错误率
树深度
树数目:1
树数目:50
树数目:100
树数目:500

RF scikit-learn相关参数

参数	RandomForestClassifier	RandomForestRegressor
criterion	指定划分标准，默认为gini，不支持其它参数	指定划分标准，可选”mse”和”mae”; 默认mse
loss	不支持	指定误差的计算方式，可选参数”linear”, “square”, “exponential”, 默认为”linear”；一般不用改动
n_estimators	最大迭代次数，也就是最多允许的决策树的数目，值过小可能会导致欠拟合，值过大可能会导致过拟合，一般50~100比较适合，默认10	
max_features	给定在进行最佳特征划分的时候，选择多少个特征进行考虑；默认为auto；max_features=sqrt(n_features)；一般不建议改动，具体参数见官网文档。	
max_depth	给定树的深度，默认为None，表示一致扩展到叶子节点足够纯或者样本数小于min_samples_split	
min_samples_split	给定树构建过程中，叶子节点中最少样本数量，默认为2	
min_samples_leaf	给定每个叶子节点中，最少的样本数目是多少，默认为2	
bootstrap	是否进行有放回的重采样，默认为True	

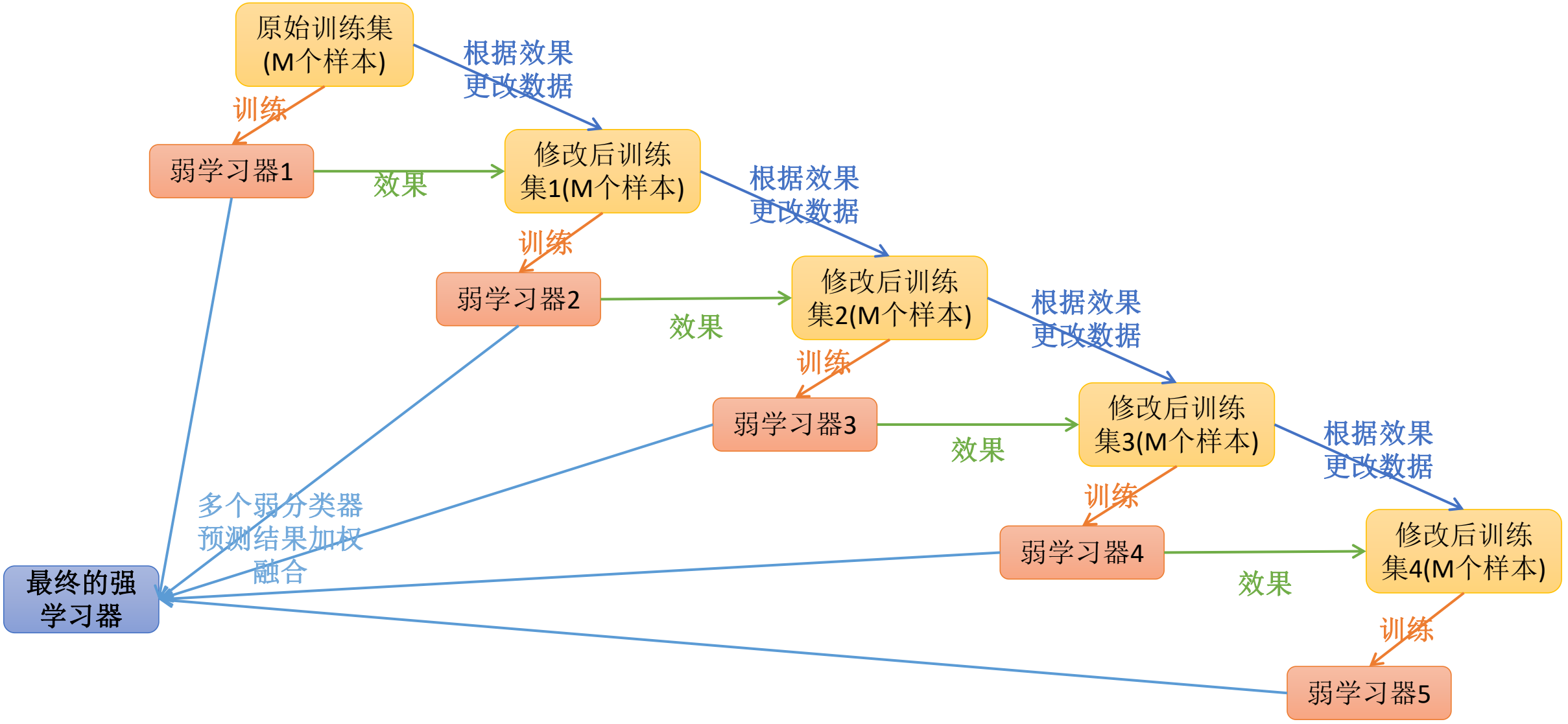
随机森林的思考

- 在随机森林的构建过程中，由于各棵树之间是没有关系的，相对独立的；在构建的过程中，构建第m棵子树的时候，不会考虑前面的m-1棵树。
- 思考：
  - 如果在构建第m棵子树的时候，考虑到前m-1棵子树的结果，会不会对最终结果产生有益的影响？
  - 各个决策树组成随机森林后，在形成最终结果的时候能不能给定一种既定的决策顺序呢？(也就是那颗子树先进行决策、那颗子树后进行决策)

Boosting

- 提升学习（Boosting）是一种机器学习技术，可以用于**回归**和**分类**的问题，它每一步产生**弱预测模型**(如决策树)，并**加权累加**到总模型中；如果每一步的弱预测模型的生成都是依据损失函数的梯度方式的，那么就称为梯度提升(Gradient boosting)；
- 提升技术的意义：如果一个问题存在**弱预测模型**，那么可以通过提升技术的办法得到一个**强预测模型；**
- 常见的模型有：
  - Adaboost
  - Gradient Boosting(GBT/GBDT/GBRT)

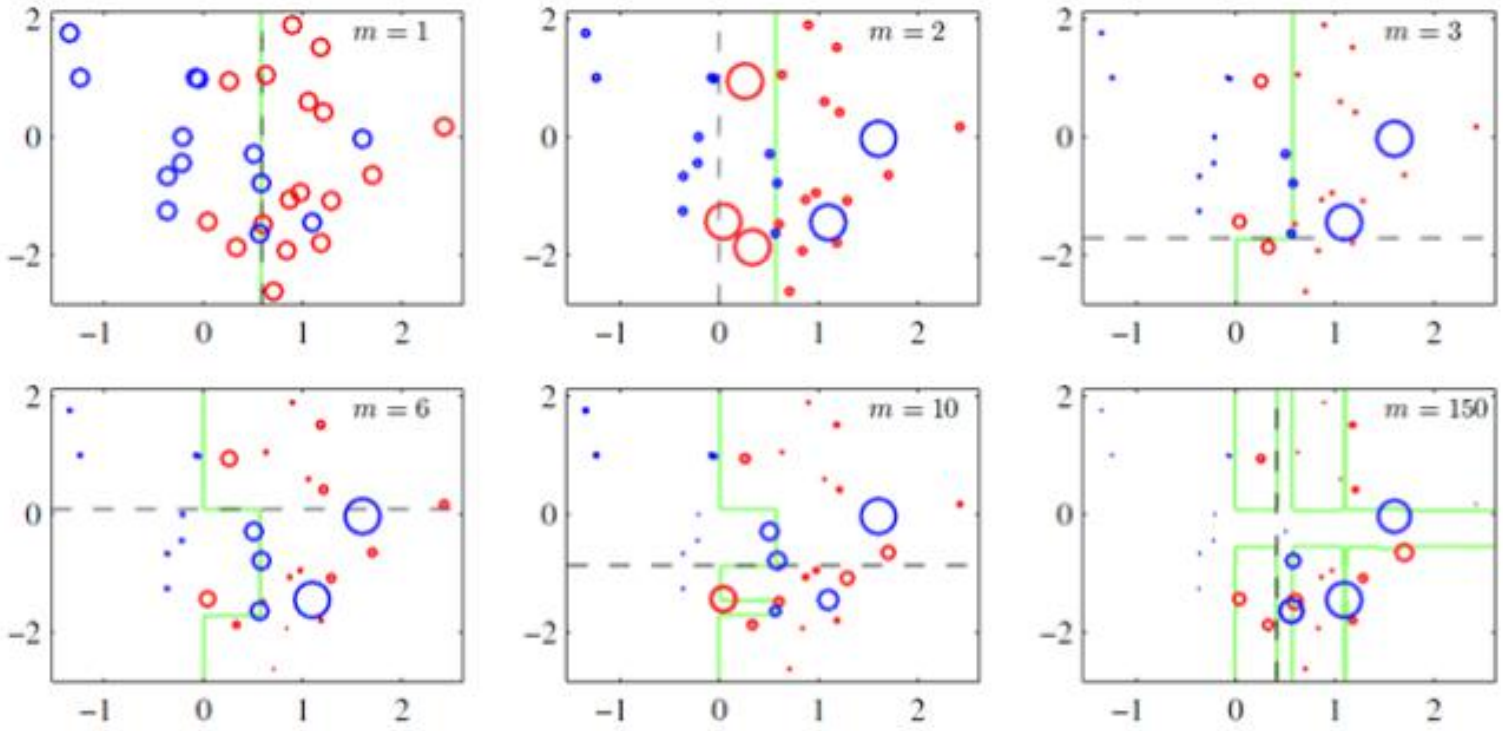
Boosting


原始训练集
(M个样本)
根据效果
更改数据
训练
弱学习器1
效果
修改后训练
集1(M个样本)
根据效果
更改数据
训练
弱学习器2
效果
修改后训练
集2(M个样本)
根据效果
更改数据
训练
弱学习器3
效果
修改后训练
集3(M个样本)
根据效果
更改数据
训练
弱学习器4
效果
修改后训练
集4(M个样本)
训练
弱学习器5
多个弱分类器
预测结果加权
融合
最终的强
学习器

AdaBoost算法原理

- Adaptive Boosting是一种迭代算法。每轮迭代中会在训练集上产生一个新的学习器，然后使用该学习器对所有样本进行预测，以评估每个样本的重要性(Informative)。换句话来讲就是，算法/子模型会为每个样本赋予一个权重，每次用训练好的学习器标注/预测各个样本(训练数据)，如果某个样本点被预测的越正确，则将样本权重降低；否则提高样本的权重。权重越高的样本在下一个迭代训练中所占的权重就越大，也就是说越难区分的样本在训练过程中会变得越重要；
- 整个迭代过程直到错误率足够小或者达到一定的迭代次数为止。

样本加权



Adaboost算法

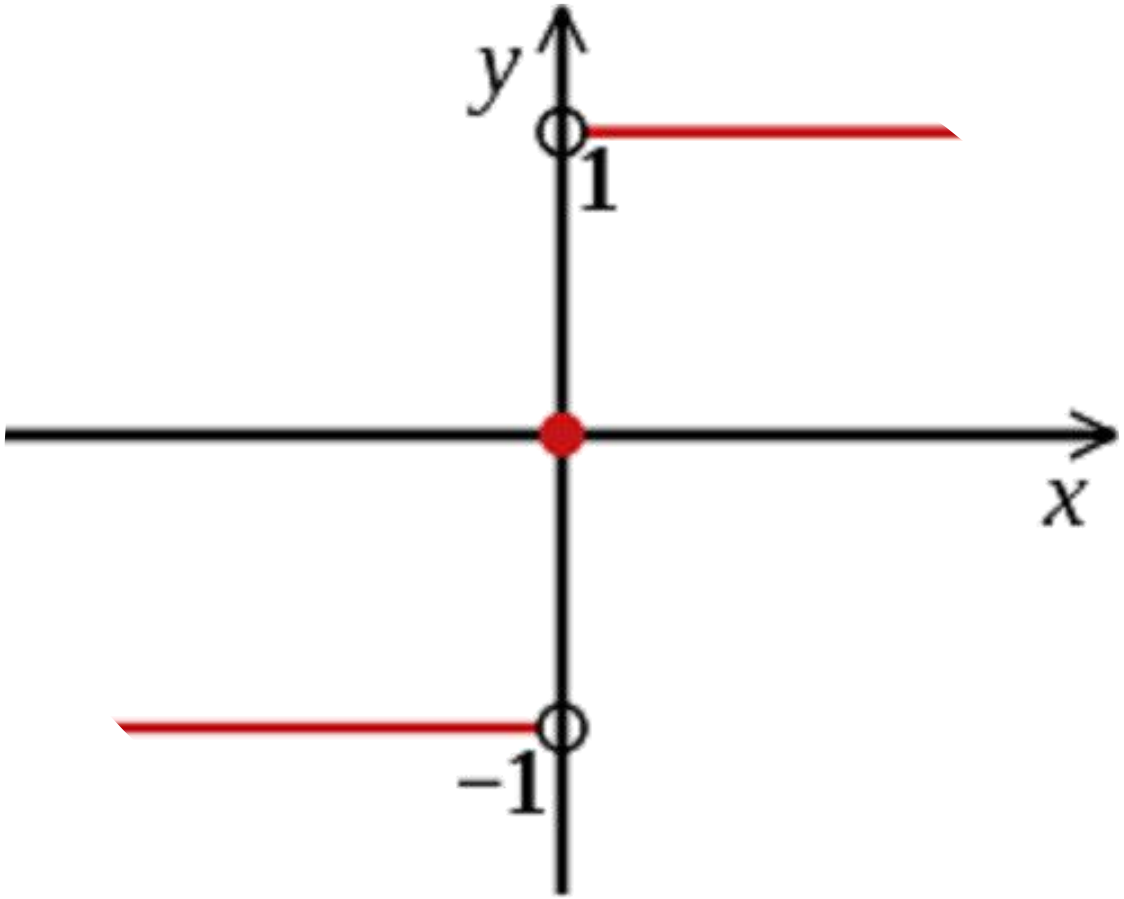
- Adaboost算法将基分类器的线性组合作为强分类器，同时给分类误差率较小的基本分类器以大的权值，给分类误差率较大的基分类器以小的权重值；构建的线性组合为：

$$f(x)=\sum_{m=1}^{M}\alpha_m G_m(x)$$

- 最终分类器是在线性组合的基础上进行Sign函数转换：

$$G(x)=sign(f(x))=sign\left[\sum_{m=1}^{M}\alpha_m G_m(x)\right]$$

Sign函数



AdaBoost算法原理

- 最终的强学习器：$$G(x)=sign(f(x))=sign\left[\sum_{m=1}^{M}\alpha_m G_m(x)\right]$$
- 损失函数(以错误率作为损失函数)：$$loss=\frac{1}{n}\sum_{i=1}^{n}I(G(x_i)\neq y_i)$$
- 损失函数：$$loss=\frac{1}{n}\sum_{i=1}^{n}I(G(x_i)\neq y_i)\leq\frac{1}{n}\sum_{i=1}^{n}e^{(-y_i f(x))}$$

AdaBoost算法原理

$$loss = \frac{1}{n}\sum_{i=1}^{n} e^{(-y_i f(x_i))}$$

- 第k-1轮的强学习器：

$$f_{k-1}(x) = \sum_{j=1}^{k-1} \alpha_j G_j(x)$$

- 第k轮的强学习器：

$$f_k(x) = \sum_{j=1}^{k} \alpha_j G_j(x) \qquad f_k(x) = f_{k-1}(x) + \alpha_k G_k(x)$$

- 损失函数：$loss(\alpha_m, G_m(x)) = \frac{1}{n}\sum_{i=1}^{n} e^{(-y_i (f_{m-1}(x) + \alpha_m G_m(x)))}$

AdaBoost算法原理

$$loss(\alpha_m, G_m(x)) = \frac{1}{n}\sum_{i=1}^{n} e^{(-y_i(f_{m-1}(x)+\alpha_m G_m(x)))}$$

$$= \frac{1}{n}\sum_{i=1}^{n} e^{-y_i f_{m-1}(x)} e^{(-y_i \alpha_m G_m(x))}$$

$$\xrightarrow{令 \overline{w}_{mi} = e^{-y_i f_{m-1}(x)}}$$

$$= \frac{1}{n}\sum_{i=1}^{n} \overline{w}_{mi} e^{(-y_i \alpha_m G_m(x))}$$

AdaBoost算法原理

- 使下列公式达到最小值的α_m和G_m就是AdaBoost算法的最终解

$$loss(\alpha_m, G_m(x)) = \frac{1}{n}\sum_{i=1}^{n} \overline{w}_{mi} e^{(-y_i \alpha_m G_m(x))}$$

- G这个分类器在训练的过程中，是为了让误差率最小，所以可以认为G越好其实就是误差率越小。

$$G_m^*(x) = \min_{G_m(x)} \frac{1}{n}\sum_{i=1}^{n} \overline{w}_{mi} I(y_i \neq G_m(x_i)) \quad \varepsilon_m = P(G_m(x) \neq y) = \frac{1}{n}\sum_{i=1}^{n} \overline{w}_{mi} I(y_i \neq G_m(x_i))$$

- 对于α_m而言，通过求导然后令导数为零，可以得到公式（log对象可以以e为底也可以以2为底）：

$$\alpha_m^* = \frac{1}{2}\ln\left(\frac{1-\varepsilon_m}{\varepsilon_m}\right)$$

Adaboost算法构建过程一

- 1. 假设训练数据集T={$(X_1,Y_1),(X_2,Y_2)....(X_n,Y_n)$}
- 2. 初始化训练数据权重分布

$$D_1 = (w_{11}, w_{12}, ...w_{1i}..., w_{1n}), \quad w_{1i} = \frac{1}{n}, \quad i = 1,2,...,n$$

- 3. 使用具有权值分布D_m的训练数据集学习，得到基本分类器

$$G_m(x): x \to \{-1,+1\}$$

- 4. 计算$G_m(x)$在训练集上的分类误差

$$\varepsilon_m = P(G_m(x_i) \neq y_i) = \sum_{i=1}^{n} w_{mi} I(G_m(x_i) \neq y_i)$$

- 5. 计算$G_m(x)$模型的权重系数α_m：

$$\alpha_m = \frac{1}{2} * \log_2\left(\frac{1-\varepsilon_m}{\varepsilon_m}\right)$$

Adaboost算法构建过程二

- 6. 权重训练数据集的权值分布

$$D_{\mathrm{m+1}} = \left(w_{m+1,1}, w_{m+1,2}, \ldots w_{m+1,i} \ldots, w_{m+1,n}\right) \quad w_{m+1,i} = \frac{w_{m,i}}{Z_m} e^{-\alpha_m y_i \mathrm{G}_m(x_i)}$$

- 7. 这里Z_m是规范化因子(归一化)

$$Z_m = \sum_{i=1}^{n} w_{m,i} e^{-\alpha_m y_i \mathrm{G}_m(x_i)}$$

- 8. 构建基本分类器的线性组合

$$f(x) = \sum_{m=1}^{M} \alpha_m G_m(x)$$

- 9. 得到最终分类器

$$G(x) = sign(f(x)) = sign\left(\sum_{m=1}^{M} \alpha_m G_m(x)\right)$$

Adaboost算法的直观理解

- 使用下列样本作为训练数据，试图使用AdaBoost算法学习一个强分类器

序号	1	2	3	4	5	6	7	8	9	10
X	0	1	2	3	4	5	6	7	8	9
Y	1	1	1	-1	-1	-1	1	1	1	-1

- 初始化训练数据集的权值分布

$$D_1 = (w_{11}, w_{12}, \ldots w_{1i} \ldots, w_{1n}), \quad w_{1i} = \frac{1}{N}, \quad i = 1,2,\ldots, N$$

$$w_{1i} = 0.1$$


北风网
IBEIFENG.COM