

集成算法介绍

Ensemble

吴婷

GitHub: [hallo128](#)

题纲

- 为什么需要使用集成算法
- 常用的三种集成算法介绍
- 实现集成及效果说明（R语言）
- 结论和使用建议

一、为什么需要使用集成算法

- **表现效果：** 集成算法表现通常都比单个算法优异，在各种比赛中独占鳌头
- **理论支持：** 在PCA框架下，弱可学习与强可学习是等价的。但发现弱学习器通常比发现强学习算法容易得多。

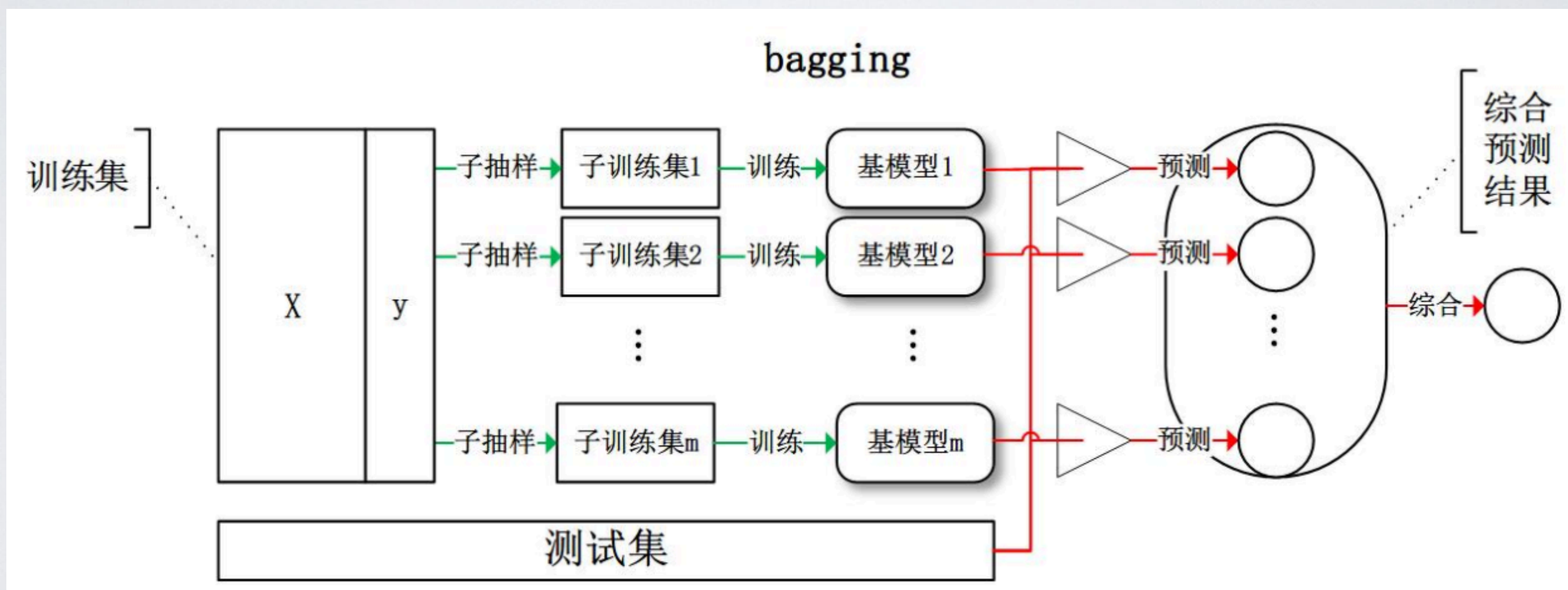
注：

- PCA: probably approximately correct(近似概率正确)

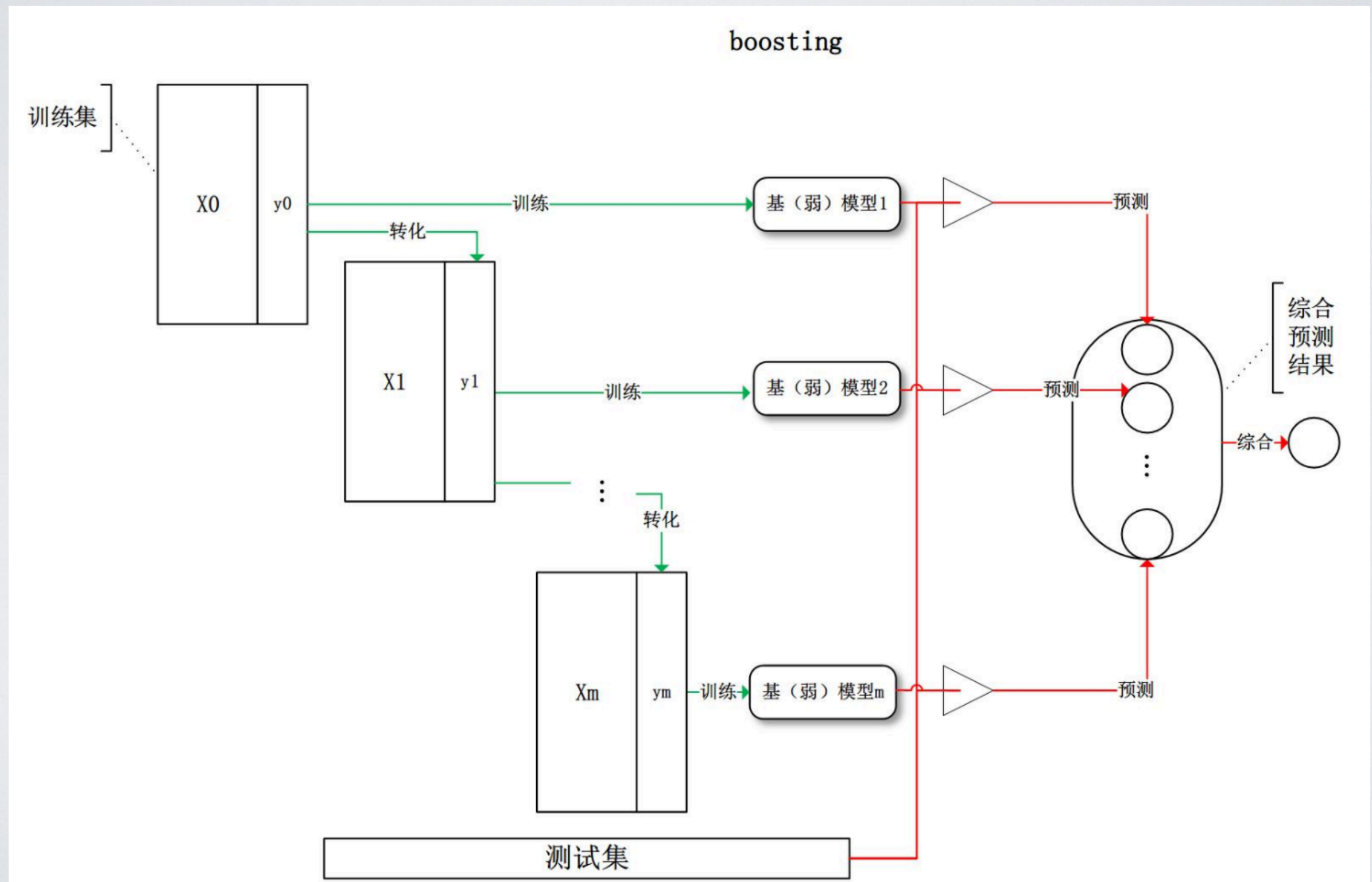
二、常用的三种集成算法介绍

Bagging(装袋)、Boosting(提升)、Stacking(堆叠)

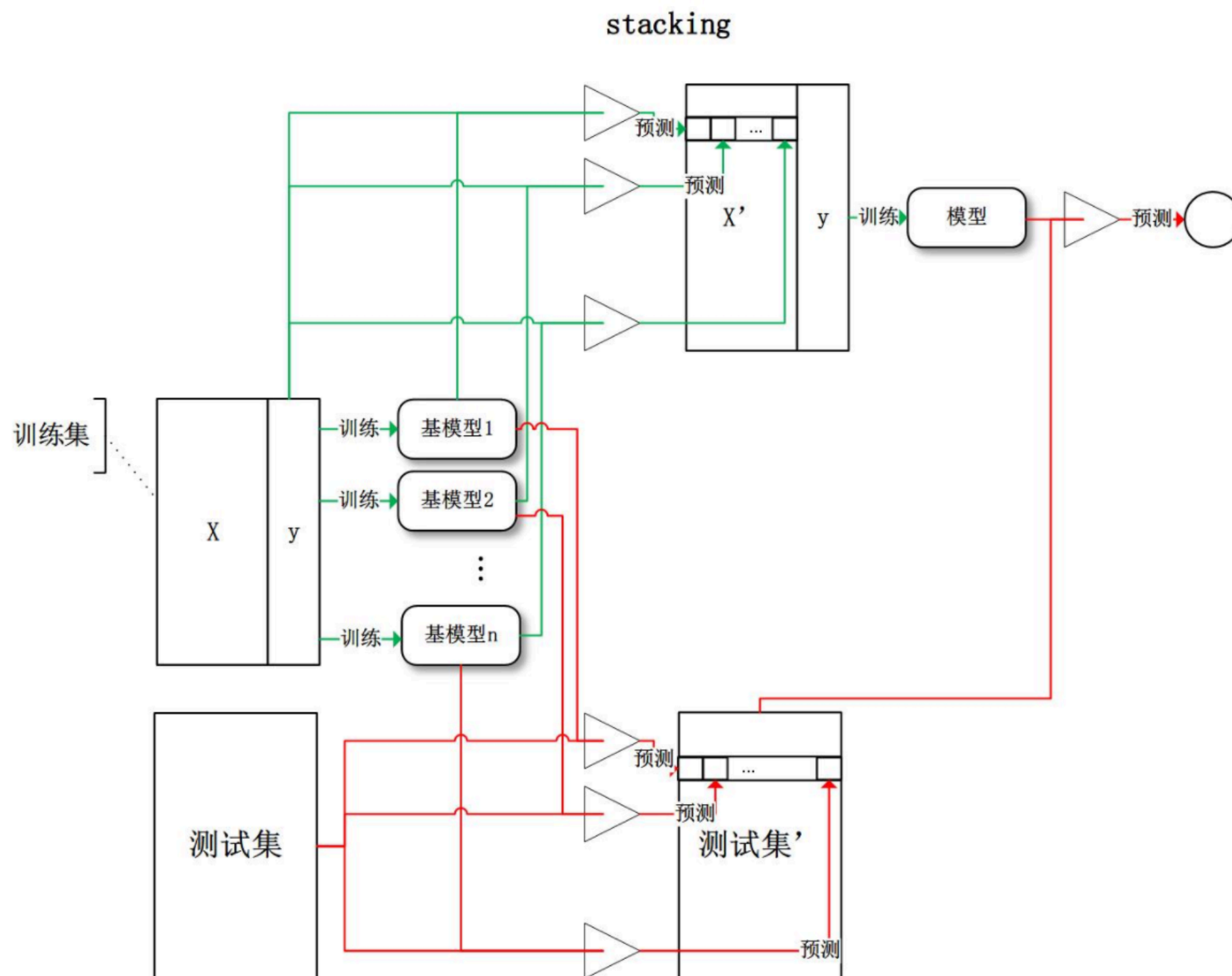
Bagging: 从训练集中进行子抽样组成每个基模型所需要的子训练集, 对所有基模型预测的结果进行综合产生最终的预测结果



Boosting: 训练过程为阶梯状，基模型按次序一一进行训练（实现上可以做到并行），基模型的训练集按照某种策略每次都进行一定的转化。对所有基模型预测的结果进行线性综合产生最终的预测结果



Stacking: 将训练好的所有基模型对训练基进行预测，第j个基模型对第i个训练样本的预测值将作为新的训练集中第i个样本的第j个特征值，最后基于新的训练集进行训练。同理，预测的过程也要先经过所有基模型的预测形成新的测试集，最后再对测试集进行预测



	Bagging	Boosting
基模型特点	<ul style="list-style-type: none">强模型(低偏差, 高方差)模型间不相关	<ul style="list-style-type: none">弱模型(高偏差, 低方差)顺序生成, 相关
集成目标	降低方差	降低偏差
解决思路	综合平均降低方差	对上一次结果进行修正 (如: 修正样本判错权重、 对上一轮的残差进行拟合)
关键	<ul style="list-style-type: none">如何构建有差异性的分类器	<ul style="list-style-type: none">权重修正设定损失函数的定义
应用	随机森林 是对一般Bagging的改进(行列随机)	GBRT、GBDT、AdaBoost、 GBM、XGBoost

三、实现集成及效果说明

- 准备工作：在R上安装“mlr”包；打开mlr官网
- 实验流程：先直接使用基本算法+集成算法，查看判别效果；再对集成算法进行调试参数，再比较判别效果

- 运行文件说明：

mlr_demo3.R：包含了所有尝试过的模型调试内容

mlr_二分类最优I.R：表现较好的二分类模型，用于集成效果说明

注：

- 具体运行文件所在GitHub：
- mlr使用说明：<https://mlr-org.github.io/mlr-tutorial/release/html/index.html>

Classification (84)

For classification the following additional learner properties are relevant and shown in column **Props**:

- *prob*: The method can predict probabilities,
- *oneclass*, *twoclass*, *multiclass*: One-class, two-class (binary) or multi-class classification problems be handled,
- *class.weights*: Class weights can be handled.

Class / Short Name / Name	Packages	Num.	Fac.	Ord.	NAs	Weights	Props	Note
classif.ada <i>ada</i> ada Boosting	ada rpart	X	X				prob twoclass	<code>xval</code> has been set to <code>0</code> by default for speed.
classif.adaboostm1 <i>adaboostm1</i> ada Boosting M1	RWeka	X	X				prob twoclass multiclass	NAs are directly passed to WEKA with <code>na.action = na.pass</code> .
classif.bartMachine <i>bartmachine</i> Bayesian Additive Regression Trees	bartMachine	X	X		X		prob twoclass	<code>use_missing_data</code> has been set to <code>TRUE</code> by default to allow missing data support.
classif.binomial <i>binomial</i> Binomial Regression	stats	X	X			X	prob twoclass	Delegates to <code>glm</code> with freely choosable binomial link function via learner parameter <code>link</code> . We set 'model' to FALSE by default to save memory.
classif.blackboost <i>blackboost</i> Gradient Boosting With Regression Trees	mboost party	X	X		X	X	prob twoclass	See <code>?ctree_control</code> for possible breakage for nominal features with missingness. <code>family</code> has been set to <code>Binomial</code> by default. For 'family' 'AUC' and 'AdaExp' probabilities cannot be predcited.

mlr根据学习任务情况给出的学习器建议

class	名称	package	只适用于二分类	集成算法
classif.binomial	"Binomial Regression"	stats	1	
classif.blackboost	"Gradient Boosting With Regression Trees"	mboost,party	1	boost
classif.boosting	"Adabag Boosting"	adabag,rpart		boost
classif.cforest	"Random forest based on conditional inference"	party		bag
classif.ctree	"Conditional Inference Trees"	party		
classif.earth	"Flexible Discriminant Analysis"	earth,stats		
classif.featureless	"Featureless classifier"	mlr		
classif.gamboost	"Gradient boosting with smooth components"	mboost	1	boost
classif.gausspr	"Gaussian Processes"	kernlab		
classif.gbm	"Gradient Boosting Machine"	gbm		boost
classif.glmboost	"Boosting for GLMs"	mboost	1	boost
classif.kknn	"k-Nearest Neighbor"	kknn		
classif.ksvm	"Support Vector Machines"	kernlab		
classif.lda	"Linear Discriminant Analysis"	MASS		
classif.logreg	"Logistic Regression"	stats	1	
classif.multinom	"Multinomial Regression"	nnet		
classif.naiveBayes	"Naive Bayes"	e1071		
classif.nnet	"Neural Network"	nnet		
classif.probit	"Probit Regression"	stats	1	
classif.qda	"Quadratic Discriminant Analysis"	MASS		
classif.randomForest	"Random Forest"	randomForest		bag
classif.rpart	"Decision Tree"	rpart		
classif.svm	"Support Vector Machines (libsvm)"	e1071		
classif.xgboost	"eXtreme Gradient Boosting"	xgboost		boost

尝试后发现，GBRT和AdaBoost运行调试耗时太长。

最后考虑的集成算法有：GBM、RF、XGBoost

三、实现集成及效果说明

- 任务：垃圾邮件的二分类检测任务
- 适用于：自变量X为数值型；输出概率值；平衡样本
- 集成算法：不仅考虑指标表现，还要考虑运行速度

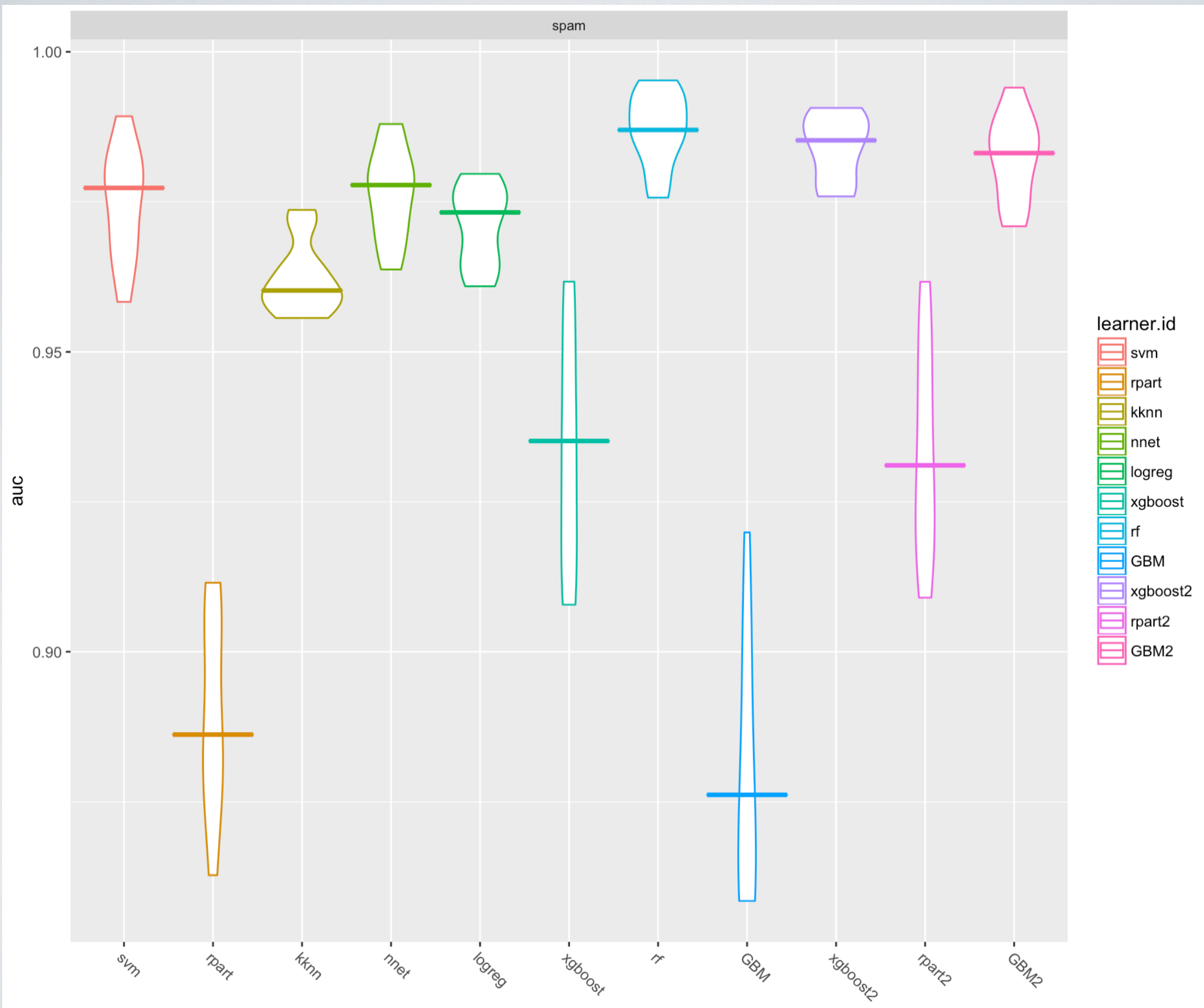
	task.id	learner.id	logloss.test.mean	auc.test.mean	ssr.test.mean	timetrain.test.mean
1	spam	svm	0.1928965	0.9747590	0.9426320	2.6064
2	spam	rpart	0.3452475	0.8897688	0.8956389	0.0827
3	spam	kknn	0.6444000	0.9623043	0.9301876	0.0003
4	spam	nnet	0.1955470	0.9763317	0.9439794	0.8473
5	spam	logreg	0.2465715	0.9710861	0.9359583	0.3432
6	spam	xgboost	0.5098200	0.9333406	0.8302262	0.0451
7	spam	rf	0.1540522	0.9871119	0.9595964	9.6740
8	spam	GBM	0.6406436	0.8797528	0.7419429	0.4362

1. 算法运行速度

从上图的10折交叉验证结果可以看出：

最快的是k近邻算法；速度较慢的是：svm和rf算法。

结论：xgboost确实是集成算法中速度很快的算法。



XGBoost与GBDT算法的联系与区别

- 都是基于CART、线性分类器，相当于带L1和L2正则化项的逻辑回归(分类)和线性回归(回归)
- 优化时，xgboost对代价函数进行了二阶泰勒展开，**同时用到了一阶和二阶导**。同时xgboost支持**自定义代价函数/优化目标** (GBM算法一定程度也可以定义优化目标)
- 在代价函数里加入了**正则项**(如：树的叶子节点个数等)，用于控制模型复杂度。
- **列采样**，借鉴随机森林
- 可以处理缺失
- 支持**并行计算**，近似直方图算法
- 优化目标同时，进行了预剪枝，对节点值进行了平滑

集成算法比较[https://www.jianshu.com/p/a69194bdab9e?](https://www.jianshu.com/p/a69194bdab9e?utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommendation)

[utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommendation](https://www.jianshu.com/p/a69194bdab9e?utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommendation)

四、结论和使用建议

- 集成算法有更大的提升空间，但单个算法几乎很难提升太多
- 使用集成算法大部分时候都需要进行调参，对模型理解要求比较高
- 进行调参时，指定参数及其范围，还有寻优算法，都有很大影响。这也是使用集成算法不得不考虑的事情。
- 同时，目前使用最多的基模型就是树模型，所以要理解集成算法，不得不先掌握有关决策树的理论内容
- Stacking虽然没有重点介绍，但它很有可能为你的模型带来进一步提升。
- 模型融合

二分类算法建议（基于本次实验假定下）：

- 根据速度和结果，建议集成算法选用：GBM(调参)、RF(不调参)、XGBoost(调参)——“mlr包也会给出参数建议”
- 需要调节的参数及范围，如代码所示
- mlr重点在乎效果评判，而各个算法都有自己特别的部分，建议用mlr得到最优算法和参数后，再使用各自算法包，查看更多内容（如：随机森林-各个变量重要性衡量、袋外误差估计OOB error）

参考文献

- 李航. 统计学习方法[M]. 清华大学出版社, 2012.
- kaggle第一名的集成思路: <https://www.kaggle.com/c/otto-group-product-classification-challenge/discussion/14335>
- 集成理论: <https://cloud.tencent.com/developer/article/1031762>
- 使用sklearn进行集成学习——理论: <https://www.cnblogs.com/jasonfreak/p/5657196.html>
- 集成算法比较https://www.jianshu.com/p/a69194bdab9e?utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommendation

在本次介绍中，并没有具体讲述每个集成算法的理论部分。虽然集成的思想很简单，但要做到自己从头构建一个大部分时候表现优异，并且能快速运行的集成算法，也是很困难。个人感觉，在调参数的时候，就已经需要理解大部分的底层实现。所有集成算法可能在一遍调参数，一遍从理论模型上进行调试。更进一步的可能需要去查看源代码是怎么实现的。