**Task1任务内容：随机森林算法梳理**

**1、集成学习的概念**

集成学习(ensemble learning)是通过构建并结合多个学习器来完成学习任务，有时也被称为多分类器系统(multi-classifier system)、基于委员会的学习(committee-based learning)等。是使用多种学习算法来获得比单独使用任何单独的学习算法更好的预测性能。

目前集成学习可分为两类：

（1）个体学习器间存在强依赖关系，必须串行生成的序列化方法，主要代表是：Boosting

（1）个体学习器间不存在强依赖关系，可同时生成的并行化方法，主要代表是：Bagging和随机森林(Random Forest)。

**2、个体学习器的概念：**

个体学习器通常由一个现有的学习算法从训练数据产生，如决策树算法、BP神经网络算法等。只包含同种类型的个体学习器的集成是“同质”的，同质集成中的个体学习器称为“基学习器”，相应的算法称为“基学习算法”。异质集成中的个体学习器一般称为“组建学习器”或者直接称为个体学习器。

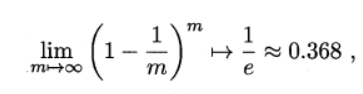
**3、Boosting/bagging的概念、异同点：**

（1）Boosting是一族将弱学习器提升为强学习器的算法。

工作机制：从初始训练集训练出一个基学习器，再根据基学习器的表现对训练样本分布进行调整，使得先前基学习器做错的训练样本在后续受到更多的关注，然后基于调整后的样本分布来训练下一个基学习器；如此重复进行，直至基学习器数目达到事先指定的值T，最终将这T个基学习器进行加权结合。（西瓜书P173页，蔡天文集成学习PPT）

（2）bagging的介绍

在介绍bagging之前首先介绍“自助法”(Bootstrapping)，该方法直接以自助采样法(bootstrap sampling)为基础，在对于给定m个样本的数据集D，每次从D中随机挑选一个样本，将其拷贝进D’中，然后再将该样本放回至D中，这样该样本在下次仍然可能会被采集；重复执行m次，就得到了包含m个样本的数据集D’，D中m次采集中始终不被采集到的概率为（1-1/m）^m，取极限得到：



可见D中有36.8%的样本未出现在采集数据集D’中，将D’作为训练集，D\D’作为测试集，这样的测试结果称为“包外估计”(out-of-bag estimate)。

自助法好处：在数据集较小，难以有效划分训练/测试集时很有用；从初始集中产生多个不同的测试集，这对集成学习等方法有很大好处。

坏处：产生的数据集改变了初始数据集的分布，会引入估计偏差。因此在初始数据集足够时，留出法和交叉验证法更常用。（西瓜书P28页）

Bagging是并行式集成学习方法的最著名代表。它是直接基于自助采样法的。按照自助采样法，我们可以采集出T个含m个训练样本的采样集，然后基于每个采样集训练出一个基学习器，再将基学习器进行结合，这就是bagging的流程。

Bagging对分类任务使用简单投票法，对回归任务使用简单平均法。若分类预测时出现两个类收到同样票数的情形，最简单的做法是随机选一个，也可以考察学习器投票的置信度来确定最终胜者。（西瓜书P178页）

（3）Boosting和Bagging的异同点(参考链接：https://blog.csdn.net/mr\_tyting/article/details/72957853)

Boosting是个体学习器间存在强依赖关系，必须串行生成的序列化方法；Bagging是个体学习器间不存在强依赖关系，可同时生成的并行化方法。

* 样本选择上：Bagging：训练集是在原始集中有放回选取的，从原始集中选出的各轮训练集之间是独立的。Boosting：每一轮的训练集不变，只是训练集中每个样例在分类器中的权重发生变化。而权值是根据上一轮的分类结果进行调整。
* 样例权重：Bagging：使用均匀取样，每个样例的权重相等。Boosting：根据错误率不断调整样例的权值，错误率越大则权重越大。
* 预测函数：Bagging：所有预测函数的权重相等。Boosting：每个弱分类器都有相应的权重，对于分类误差小的分类器会有更大的权重。
* 并行计算：Bagging：各个预测函数可以并行生成。Boosting：理论上各个预测函数只能顺序生成，因为后一个模型参数需要前一轮模型的结果。计算角度来看，两种方法都可以并行。bagging, random forest并行化方法显而意见。boosting有强力工具stochastic gradient boosting。
* bagging是减少variance，而boosting是减少bias。在机器学习中，我们用训练数据集去训练（学习）一个model（模型），通常的做法是定义一个Loss function（误差函数），通过将这个Loss（或者叫error）的最小化过程，来提高模型的性能（performance）。然而我们学习一个模型的目的是为了解决实际的问题（或者说是训练数据集这个领域（field）中的一般化问题），单纯地将训练数据集的loss最小化，并不能保证在解决更一般的问题时模型仍然是最优，甚至不能保证模型是可用的。这个训练数据集的loss与一般化的数据集的loss之间的差异就叫做generalization error。而generalization error又可以细分为Bias和Variance两个部分。即error=Bias+Variance。
* 过拟合与欠拟合。Bagging里面每个分类器是强分类器，因为他降低的是方差，方差过高需要降低是过拟合。boosting里面每个分类器是弱分类器，因为他降低的是偏差，偏差过高是欠拟合。

**4、理解不同的结合策略(平均法，投票法，学习法)**

学习器结合可以带来的三大好处：

(1)统计的原因

结合多个学习器可以减少单学习器因误选而导致泛化性能不佳的风险。

(2)计算的原因

通过多次运行之后进行结合，可以降低陷入糟糕局部极小点的风险。

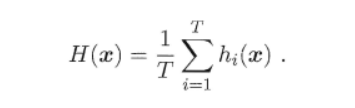
(3)表示的原因

某些学习任务的真实假设可能不在当前学习算法所考虑的假设空间中，此时若使用单学习器肯定无效，通过结合多个学习器，相应的假设空间有所扩大，有可能学得更好的近似。

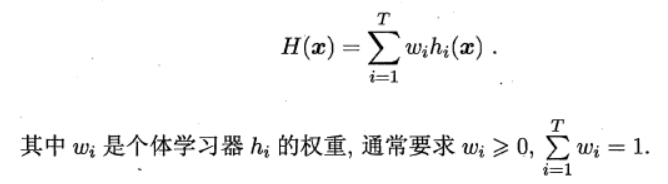
假设集成包含T个基学习器{h1，h2，…，hT}，其中hi在示例x上的输出为hi(x)，下面介绍几种常见策略。

(1)平均法

**简单平局法(simple averaging)**



**加权平均法(weighted averaging)**

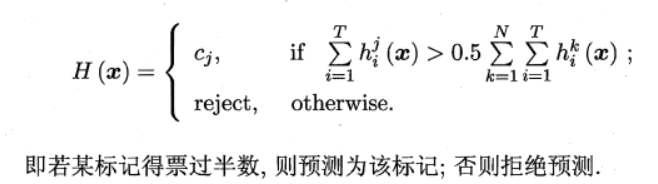


一般而言，在个体学习器性能相差较大时宜使用加权平均法，而在个体学习器性能相近时宜使用简单平均法。

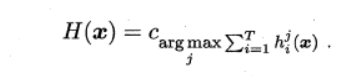
(2) 投票法

学习器h\_i将从类别标记集合{c\_1，c\_2，…，c\_N}中预测出一个标记，h\_i在样本x上的预测输出表示为一个N维向量(h\_i^1(x); h\_i^2(x);…; h\_i^N(x))，其中h\_i^j(x)是h\_i在类别标记c\_j上的输出。

**绝对多数投票法(majority voting)**

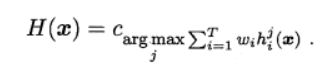


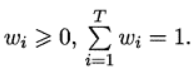
**相对多数投票法(plurality voting)**



即预测为得票最多的标记，若同时有多个标记获得最高票，则从中随机选取一个。

**加权投票法(weighted voting)**



与加权平均法类似，w\_i是h\_i的权重，通常

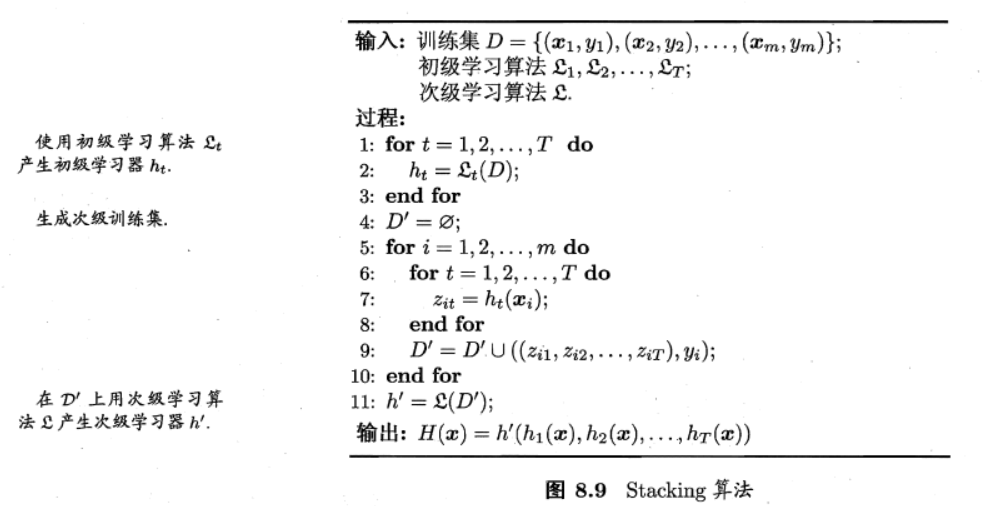
在现实任务中，不同个体学习器产生不同类型的h\_i^j(x)值，常见的有：

* 类标记：h\_i^j(x)取值为{0，1}，若h\_i将样本x预测为类别c\_j则取值为1，否则为0.使用类标记的投票称为“硬投票”(hard voting)
* 类概率：h\_i^j(x)取值为[0，1]，相当于对后验概率P(c\_j | x)的一个估计，使用类概率的投票称为“软投票”(soft voting)

(3) 学习法

通过另一个学习器来进行结合。Stacking是学习法的典型代表。我们把个体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或元学习器(meta-learner)。

Stacking先从初始数据集训练出初始学习器，然后“生成”一个新数据集用于训练次级学习器，在这个新数据集中，初级学习器的输出被当作样例输入特征，而初始样本的标记仍被当作样例标记。这里假定初级学习器使用不同学习算法产生，即初级集成是异质的。



**5、随机森林的思想**

**随机森林(Random Forest, RF)**是bagging的一个扩展变体，RF以决策树为基学习器构建Bagging集成的基础上，进一步在决策树的训练过程中引入了随机属性选择。简而言之，就是生成一片由决策树构成的森林。

**信息熵(Entropy)**：用来度量随机变量的不确定性；

**信息增益：**以某特征划分数据集前后的熵的差值，划分前样本集合D的熵是一定的 ，为entropy(前)，使用某个特征A划分数据集D，计算划分后的数据子集的熵 entropy(后)。信息增益越大，则这个特征的选择性越好，信息增益 = entropy(前) - entropy(后)。

**基尼(gini)不纯度：**在样本集合中一个随机选中的样本被分错的概率。基尼不纯度越小，则这个特征的选择性越好。

**决策树的分类：**ID3，C4.5和 CART树 (classification and regression tree)。其中，ID3和C4.5的计算和熵有关，CART树则用的是基尼不纯度。

随机森林更多用CART树，因为用基尼不纯度生成决策树比用信息增益计算工程效率更高一点。

**随机森林的生成规则：**

随机森林就是用CART决策树作为Bagging算法中的弱学习器的实现，同时在每棵树的特征选择也用到了随机且有放回抽样，所以随机森林的生成规则如下：

1 从原始训练集中随机有放回采样N个训练样本，共进行T次采样，生成T个训练集

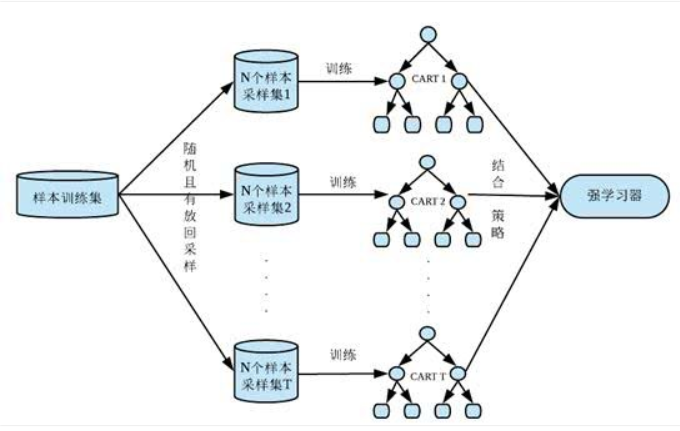
2 用T个训练集，分别训练T个CART树模型

3 如果特征维度为M，指定一个常数m

4 将生成的T棵决策树组成随机森林

5 对于分类问题，由T个CART树投票表决产生分类结果。对于回归问题，由T棵树预测结果的均值作为最终的预测结果。

所以随机森林归的随机归纳为一句话，即**数据随机采样，特征随机采样**。



**一般而言，在个体学习器性能相差较大时宜使用加权平均法，而在个体学习器性能相近时宜使用简单平均法**。

**6、随机森林的推广（参考：https://yq.aliyun.com/articles/667534）**

RF推广算法在实际应用中占有比较好的特性，应用比较广泛，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 Tree Embedding(TRTE)、Isolation Forest。

（1）Extra Tree

Extra Tree是随机森林(RF)的一个变种，原理基本和随机森林一样，区别如下：

* RF会随机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。
* 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的特征值。Extra Tree则是随机选择一个特征值来划分决策树。

由于Extra Tree是随机选择特征点进行划分，所以最后得到的决策树规模会大于RF生成的决策树，Extra Tree决策树的方差会减少，泛化能力比RF更强。

（2）Totally Random Tree 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，从而让高维的数据更好得用于分类回归模型。

TRTE的算法的转化过程类似于RF算法，建立T个决策树来拟合数据。当决策树构建完成之后，数据集里的每个数据在T个决策树中叶子节点的位置就固定下来了，将位置信息转换为向量即完成了算法的转换。

样本根据TRTE进行转化特征，最后可能得到更高维的数据，也可能得到更低维的数据，如词袋法中的特征有2000个，做完TRTE后只剩下几百个。

经过TRTE转化后的编码可以用于无监督的分类操作，将相似的特征码聚类到一起，最后完成分类的操作。

（3）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和RF的区别在于：

1、在随机采样的过程中，一般只需要少量的数据。

2、在进行决策树的构建过程中，IForest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的划分深度是比较小的，即max\_depth较小。

区分原因：目的是异常点检测，所以只要能够区分出异常即可，不需要大量的数据；此外在异常点检测过程中，一般也不需要太大规模的决策树。

下面说说 \_\_IForest\_\_为什么可以做异常点判断。

\_\_对于异常点的判断：\_\_将测试样本x拟合到T棵决策树上，计算每棵树上该样本的叶子节点的深度ht(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(x,m)的取值范围为[0,1]，越接近1，说明异常点概率越大。

**结论： 叶子节点的深度越深，异常点的概率越小。树平均深度越浅，说明异常值概率越大。**

**7、随机森林的优缺点（参考：https://yq.aliyun.com/articles/667534）**

优点：

* 训练可以并行化，对于大规模样本的训练具有速度的优势。
* 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。
* 可以给出各个特征值的重要性列表
* 由于存在随机抽样，训练出来的模型方差小，泛化能力强
* 有out-of-bag数据，不需要单独的测试集
* 对于部分特征的丢失不敏感
* 有很强的抗干扰能力，所以当数据存在大量数据缺失，用随机森林是不错选择

缺点：

* 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
* 当数据噪声较大时，依旧会产生过拟合现象

**8、随机森林在sklearn中的参数解释**

随机森林的参数大部分和决策树一样，使用sklearn.tree.RandomForestClassfier类来实现随机森林算法，python调用如下：

构造函数主要参数和意义如下：

[n\_estimators]：string类型，可选（默认为10），森林里（决策）树的数目。就是弱学习器的最大迭代次数，或者说最大的弱学习器的个数。一般来说n\_estimators太小，容易欠拟合，n\_estimators太大，计算量会太大，并且n\_estimators到一定的数量后，再增大n\_estimators获得的模型提升会很小，所以一般选择一个适中的数值，默认是10。

[criterion]：string类型，可选（默认为’gini’），即CART树做划分时对特征的评价标准。分类模型和回归模型的损失函数是不一样的。分类RF对应的CART分类树默认是基尼系数gini, 另一个可选择的标准是信息增益。回归RF对应的CART回归树默认是均方差MSE，另一个可以选择的标准是绝对值MAE。一般来说选择默认的标准就已经很好了。

[max\_features] : Int, float, string or None, 可选(默认为‘auto’), 随机森林里单个决策树的所使用的最多特征量。 显然，这个值越高，则随机森林的速度越慢。另外，该值越高，则虽仅森林中的多样性越低,同样会降低预测准确率。一般推荐这个值为总特征数量的均方根。

[max\_depth] : Int or None，可选(默认为’None’) , 设置树的最大深度，默认为None，这样建树时，会使每一个叶节点只有一个类别，或是达到min\_samples\_split。数据少或者特征少的时候可以不管这个值。如果模型样本量多，特征也多的情况下，推荐限制这个最大深度，具体的取值取决于数据的分布。常用的可以取值10-100之间。常用来解决过拟合。

[min\_impurity\_split] : Float类型。决策树在创建分支时，信息增益必须大于这个阀值，否则不分裂

[bootstrap] : 布尔型，可选(默认为‘True’)，是否有放回的采样

[oob\_score] : 布尔型，可选(默认为‘False’)。即是否采用袋外样本来评估模型的好坏。默认识False。推荐设置为True，因为袋外分数反应了一个模型拟合后的泛化能力。再加上OOB数据的存在，可以不需要交叉验证，所以可利用OOB Error作为评判m的指标。多单个模型的参数训练，我们知道可以用cross validation来进行，但是特别消耗时间，而且对于随机森林这种情况也没有大的必要，所以就用这个数据对决策树模型进行验证，算是一个简单的交叉验证。性能消耗小，但是效果不错。

[n\_jobs] : interger类型，可选(默认为1)。用于拟合和预测并行运行的工作（作业）数量。这个在ensemble算法中非常重要，尤其是bagging（而非boosting，因为boosting的每次迭代之间有影响，所以很难进行并行化），因为可以并行从而提高性能。1=不并行；n：n个并行；-1：CPU有多少核，就启动多少job。

[random\_state] : Int，RandomState instance or None(默认为’None’)。随机树生成器使用的种子，保证模型的输出具有可复制性。当它被赋于一个指定值，且模型训练具有相同的参数和相同的训练数据时，该模型将始终产生相同的结果。

[warm\_start] : 布尔型，可选(默认为‘False’)。热启动，决定是否使用上次调用该类的结果，然后增加新的。

**9、随机森林的应用场景**

* 随机森林适用于数据维数相对较低（几十维），且需要达到一定准确率的场景。
* 随机森林在多数数据集上都有不错的表现，相对来说，出现预测效果很差的情况较少。可以作为模型的baseline作为参考。

参考：

西瓜书

cs229吴恩达机器学习课程

李航统计学习

谷歌搜索

公式推导参考：http://t.cn/EJ4F9Q0