理解梯度提升算法 1-梯度提升算法

梯度提升（gradient boosting）算法是集成学习的典型实现，与 AdaBoost 同属提升算法（boosting）家族。如果弱学习器采用决策树，则称为梯度提升树（gradient boosting decision tree，简称 GBDT）。梯度提升树的改进型算法如 XGBoost、lightBGM 在数据挖掘领域得到了成功的应用，是各种算法比赛中常用的算法。对决策树的梯度提升集成可以得到高精度、高度鲁棒、可解释的算法，用于分类问题和回归问题，尤其适合对不干净数据的分析。

损失函数

有监督学习的目标从数学上看是函数拟合问题。有一个输入向量 $\mathbf{x} \in^n$，与其对应的输出变量（也称为标签值，一般是标量） y，这些变量构成一个训练样本集

$$(y_i, \mathbf{x}_i), i = 1, ..., N$$

每个样本是成对的输入和输出值 $(\mathbf{x}, y)$。机器学习的目标是找到一个最优的预测函数，它将 $\mathbf{x}$ 映射成 y，对于所有 $(\mathbf{x}, y)$，使得某一损失函数 $L(y, F(\mathbf{x}))$ 的数学期望最小化。对于有限个训练样本，数学期望是各个样本损失函数的均值

$$\min_{F(\mathbf{x})} \mathrm{E}_{y,\mathbf{x}}\left[L(y, F(\mathbf{x}))\right]$$

这个目标函数的自变量是函数 $F(\mathbf{x})$，损失函数 L 将函数映射成一个数。对于回归问题，常用的损失函数有均方误差，绝对值误差。对于分类问题，常用的是负二项式对数似然函数（logistic 回归的对数似然函数，logistic 回归的原理在 SIGAI 之前的公众号文章“理解 logistic 回归”中已经介绍），交叉熵等。

预测函数 $F(\mathbf{x})$ 的形式是不确定的，因此我们无法直接求解这个优化问题。常用的做法是对该函数的类型进行限定，称为参数化，具体化为一个函数族，如线性函数，logistic 函数，函数带有参数 $\mathbf{P}$。然后将上面的问题转化为函数优化问题，优化变量为函数的参数。用数值优化算法如梯度下降法、牛顿法、坐标下降法求解。

在机器学习中，有一类特殊的预测函数，称为加法模型

$$F(\mathbf{x}; \mathbf{P}) = \sum_{m=0}^{M} \rho_m f(\mathbf{x}; \mathbf{a}_m)$$

即要拟合的函数是多个基函数 f 的线性组合。其中模型的参数为

$$\mathbf{p} = \{\rho_m, \mathbf{a}_m\}, m = 1, ..., M$$

$\mathbf{a}_m$ 是基函数的参数，β_m 是组合系数。AdaBoost 算法使用的就是这种模型，其损失函

数为指数损失函数$\exp\left(-yF\left(\mathbf{x}\right)\right)$，求解时采用逐步优化的策略，依次确定每个基函数及其权重系数，具体原理在SIGAI之前的公众号文章“理解AdaBoost算法”中已经介绍。

加法模型的基函数$f\left(\mathbf{x};\mathbf{a}\right)$一般选用简单的函数，其参数为$\mathbf{a}$。在梯度提升树中，这些函数是决策树，如CART。对于决策树来说，参数$\mathbf{a}$为分裂变量，分裂位置，叶子节点值。决策树的原理在之前的SIGAI公众号文章“理解决策树”中已经介绍。

最速下降法

最速下降法是梯度下降法的变种。梯度下降法的原理在SIGAI之前的公众号文章“理解梯度下降法”中已经介绍。如果我们要求函数$L\left(\mathbf{x}\right)$的极小值，梯度下降法的从一个初始迭代点$\mathbf{x}_0$开始，反复沿着当前点处的负梯度方向迭代

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \rho\nabla_{\mathbf{x}}L\left(\mathbf{x}_t\right)$$

只要学习率ρ（步长）选取得当，并且还没达到驻点处，每次迭代函数值是下降的。可以证明，沿着负梯度方向，函数值下降是最快的。如果令

$$\Delta\mathbf{x}_t = -\rho\nabla_{\mathbf{x}}L\left(\mathbf{x}_t\right)$$

这称为增量。则最后得到的最优解为每次增量的累加

$$\mathbf{x}_T = \sum_{t=0}^{T}\Delta\mathbf{x}_t$$

即它是由之前每次迭代时的增量序列累加起来的。这里的学习率是人工设定的常数，最速下降法对梯度下降法的改进是学习率ρ是由算法确定的，自适应变化，如果令梯度为

$$\mathbf{g}_t = \nabla_{\mathbf{x}}L\left(\mathbf{x}_t\right)$$

则步长为下面一元函数优化问题的解

$$\rho_t = \arg\min_{\rho} L\left(\mathbf{x}_{t-1} - \rho\mathbf{g}_t\right)$$

这称为直线搜索，它沿着最速下降方向搜索最佳步长。在牛顿法中也使用了这种技术。

梯度提升算法框架

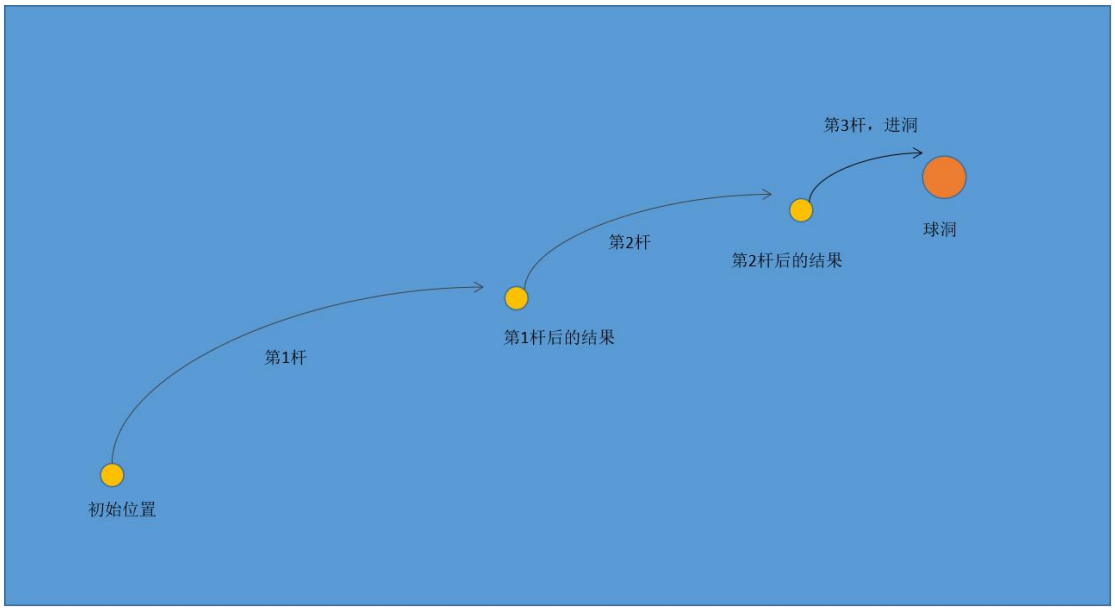
在AdaBoost算法中，求解指数损失函数的加法模型时采用的是分阶段、逐步优化的策略。依次训练每一个弱学习器，然后将它加入到已经得到的强学习器中。已经训练得到的强学习器对训练样本的输出值可以看作常数，因为指数函数的作用，加法转化为乘法，体现为样本的权重。在训练的过程中，在当前权重下训练弱学习器，然后更新样本权重。

梯度提升算法则采用了不同的思路，它不是为样本加上权重，而是在样本的标签值上或者说每次弱学习器拟合的目标值上做文章，用当前已经训练出来的强学习器$F\left(\mathbf{x}\right)$对训练样本进行预测，然后计算损失函数对$F\left(\mathbf{x}\right)$的负梯度值，如果下一个弱学习器$h\left(\mathbf{x};\mathbf{a}\right)$的预测

值指向该负梯度方向，根据梯度下降法的原理，加上这个弱学习器，即向前走一步之后损失函数值是下降的。梯度提升算法可以看做是梯度下降法与加法模型的结合。

在日常生活中，经常会遇到类似的问题，比如说打高尔夫球。刚开始，你的球离球洞有500米远，指望一杆就打进洞那是不可能的

比较现实的做法是一杆一杆的打，每一杆都让球离球洞更近一些，经过几次努力之后，球进洞了



梯度提升算法采用了这种思想-逐步求精，在前面几杆的基础上打下一杆。每打一杆时，要确定方向和力度的大小，这里的方向就是我们要拟合的弱学习器，力度就是其系数。

将$F\left(\mathbf{x}\right)$在点$\mathbf{x}$处的输出函数值看做是损失函数的自变量，最小化下面的损失函数值

$$L\left(F\left(\mathbf{x}\right)\right)=\mathrm{E}_{y,\mathbf{x}}\left[L\left(y,F\left(\mathbf{x}\right)\right)\right]$$

现在的问题变为：

已知一个样本向量$\mathbf{x}$，它的预测值$F\left(\mathbf{x}\right)$是多少的时候，损失函数可以达到最小值？我

们不能指望$F(\mathbf{x})$能一次到位，只能一步一步的改进它，最后的解是这些步的预测结果的累加

$$F(\mathbf{x})=\sum_{m=0}^{M}f_m(\mathbf{x})$$

其中$f_0(\mathbf{x})$是初始猜测值，$f_m(\mathbf{x}),m=1,...,M$为增量函数序列，对应于最速下降法中的增量$\Delta\mathbf{x}_t$。在高尔夫中，这个增量就是每次打一杆后球移动的距离。对于最速下降法，增量为

$$f_m(\mathbf{x})=-\rho_m g_m(\mathbf{x})$$

其中

$$g_m(\mathbf{x})=\left[\frac{\partial \mathrm{E}\left[L(y,F(\mathbf{x}))\right]}{\partial F(\mathbf{x})}\right]_{F(\mathbf{x})=F_{m-1}(\mathbf{x})}$$

这是损失函数对$F(\mathbf{x})$的导数值，将$F(\mathbf{x})$看做一个变量求导并且其取值为$F_{m-1}(\mathbf{x})$。其中

$$F_{m-1}(\mathbf{x})=\sum_{i=0}^{m-1}f_i(\mathbf{x})$$

这是已经训练得到的强学习器对样本的预测值，是一个常数。以高尔夫的例子来说，就是前面$m-1$杆把球打到的位置。由于积分和求导可以互换，因此有

$$g_m(\mathbf{x})=\mathrm{E}_y\left[\frac{\partial L(y,F(\mathbf{x}))}{\partial F(\mathbf{x})}\right]_{F(\mathbf{x})=F_{m-1}(\mathbf{x})}$$

步长ρ_m由直线搜索确定，即寻找使得函数值下降最快的梯度步长

$$\rho_m=\arg\min_{\rho}\mathrm{E}_{y,\mathbf{x}}\left[L(y,F_{m-1}(\mathbf{x})-\rho g_m(\mathbf{x}))\right]$$

这个步长可以理解为我们每次打一杆时用的力的大小，而负梯度则是我们用力的方向。

如果我们将弱学习器$h(\mathbf{x};\mathbf{a})$具体化，对于有限的训练样本，优化的目标为弱学习器的参数和系数

$$(\beta_m,\mathbf{a}_m)=\arg\min_{\beta,\mathbf{a}}\sum_{i=1}^{N}L(y_i,F_{m-1}(\mathbf{x}_i)+\beta f(\mathbf{x}_i;\mathbf{a}))$$

其中N为训练样本数。然后更新预测函数

$$F_m(\mathbf{x})=F_{m-1}(\mathbf{x})+\beta_m h(\mathbf{x};\mathbf{a}_m)$$

对于一般的损失函数L，直接优化上面的目标函数非常困难，只能近似求解。现在是最速下降法派上用场的时候了。如果采用最速下降法近似求解，则弱学习器拟合的目标就是负梯度，问题可以大为简化。每次迭代时先优化弱学习器，让弱学习器的输出值对准所有样本的负梯度$-g_m(\mathbf{x}_i)$方向。弱学习器参数的最优解为

$$\mathbf{a}_m = \arg\min_{\mathbf{a},\beta} \sum_{i=1}^{N} \left[-g_m(\mathbf{x}_i) - \beta h(\mathbf{x}_i;\mathbf{a}) \right]^2$$

其中N为训练样本数。即用当前弱学习器逼近负梯度值。相比于上面使用损失函数L的情况，这个问题要容易求解得多，因为这是一个二次函数，总结来说，就是用一个最小二乘问题替代了之前难以优化的函数。然后执行直线搜索

$$\rho_m = \arg\min_{\rho} \sum_{i=1}^{N} L\left(y_i, F_{m-1}(\mathbf{x}_i) + \rho h(\mathbf{x}_i;\mathbf{a}_m)\right)$$

从而确定ρ_m。注意，这一步使用的是原本的损失函数L。接下来更新预测函数

$$F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \rho_m h(\mathbf{x};\mathbf{a}_m)$$

由此得到通用的梯度提升算法框架，流程如下：

初始化强学习器$F_0(\mathbf{x}) = \arg\min_{\rho} \sum_{i=1}^{N} L(y_i, \rho)$

循环，对 m = 1, ..., M，依次训练每个弱学习器

计算伪标签值$y_i^{'} = -\left[\frac{\partial L(y_i, F(\mathbf{x}_i))}{\partial F(\mathbf{x}_i)} \right]_{F(\mathbf{x})=F_{m-1}(\mathbf{x})}, i = 1,...,N$

训练弱学习器$\mathbf{a}_m = \arg\min_{\mathbf{a},\beta} \sum_{i=1}^{N} \left[y_i^{'} - \beta h(\mathbf{x}_i;\mathbf{a}) \right]^2$

直线搜索$\rho_m = \arg\min_{\rho} \sum_{i=1}^{N} L\left(y_i, F_{m-1}(\mathbf{x}_i) + \rho h(\mathbf{x}_i;\mathbf{a}_m)\right)$

更新强学习器$F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \rho_m h(\mathbf{x};\mathbf{a}_m)$

结束循环

这里我们称样本在当前强学习器预测值点处的负梯度值为伪标签值，训练下一个弱学习器时，利用的就是这个“造出来”的标签值。因此，与 AdaBoost 一个非常根本的区别是梯度提升在训练每个弱学习器时使用的是不同的样本标签值，而 AdaBoost 使用的是不同的额样本权重。将梯度提升框架用各种不同的损失函数，得到各种具体的梯度提升算法，解决分类和回归问题。如果弱学习器是决策树，则为梯度提升树。这些具体的算法将在下一篇文章中讲述。