提升方法

Xiachong Feng

一、提升方法与AdaBoost介绍

提升方法（Boosting），是一种可以用来减小监督学习中偏差的机器学习元算法（所谓元算法，指的是“学习算法的算法”）。面对的问题是迈可·肯斯（Michael Kearns）提出的：**一组“弱学习者”的集合能否生成一个“强学习者”？**弱学习者一般是指一个分类器，它的结果只比随机分类好一点点；强学习者指分类器的结果非常接近真值。

提升方法的思路是综合多个分类器，得到更准确的分类结果。即“三个臭皮匠，顶个诸葛亮”。

提升方法的代表性算法是**AdaBoost算法**。

对于分类问题来说，给定一个训练集样本，求比较粗糙的分类规则（弱分类器）要比求精确地分类规则（强分类器）容易的多。提升方法就是从弱学习算法出发，反复学习，得到一系列弱分类器（基本分类器），然后组合这些弱分类器，构成一个强分类器。

下面有三个小问题:

（1）**如何学习一系列的弱分类器呢？**大多数提升方法都是改变训练数据的概率分布（训练数据的权值分布），针对不同的训练数据分布学习分类器。这里的权值分布不同，不是说训练数据分布不同，数据不同。而是指在计算损失时，不同数据对于损失的权重是不一样的，例如，有的数据分类错了，损失为$Loss_1$，有的数据分类错了，损失为$Loss_2$。

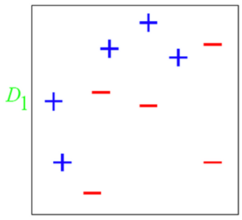
（2）**如何在每一轮修改训练数据的权重呢？**在AdaBoost中，提高那些被前一轮弱分类器错误分类样本的权值，降低那些被正确分类样本的权值。因此，在前一轮弱分类器分类错误的数据，在后一轮由于权重变大，更加受到分类器关注。

（3）**如何将弱分类器组合成一个强分类器呢？**在AdaBoost中，采用“加权多数表决法”，具体的，加大分类误差率小的弱分类器的权值，使其在表决中起主要作用；减小分类误差率大的弱分类器的权值，使其在表决中起较小作用。

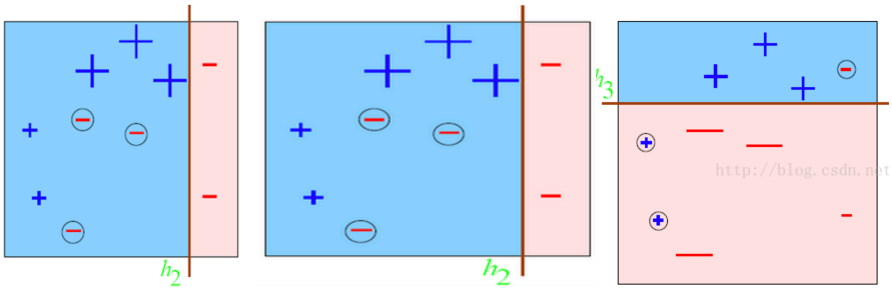
二、例子

下面举一个例子来说明将弱分类器组合为强分类器以后的优点。“+”和“-”分别表示两种类别，在这个过程中，使用水平或者垂直的直线作为分类器。

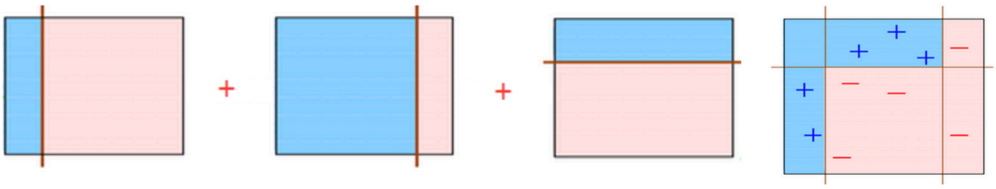
我们想要将蓝色“+”和红色“-”分开。显然，通过一条垂直的或者水平的线是不可能的。



例如使用以下三种方法，都无法达到最后分类的目的。



但是如果将上面三种方法结合，就可以得到：



通过这个例子可以直觉上感受到，通过组合弱分类器可以得到强分类器。

三、AdaBoost算法

输入：

（1） 训练数据集$T = \{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\}$，其中$x_i \in \mathcal{X} \subseteq R^n$，标记$y \in Y = \{-1, +1\}$

（2） 弱学习算法

输出：

最终分类器$G(x)$

（1）初始化训练数据的权值分布D_1，代表第一轮权值。假设训练数据集具有均匀的权值分布，即每个样本在基本分类器的学习中作用相同。

$$D_1 = (w_{11}, \dots, w_{1i}, \dots w_{1N}), w_{1i} = {}^1\!/_N, i = 1,2, \dots, N$$

（2）对$m = 1,2, \dots M$，在m轮反复学习基本分类器$G_m(x)$，具体步骤如下：

(a) 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：

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

（b）计算$G_m(x)$在训练数据集上的分类误差率，w_{mi}表示第m轮，第i个实例的权值，满足约束条件$\sum_{i=1}^{n} w_{mi}=1$，由下式可知，$G_m(x)$在加权的训练数据集上的分类误差率是被$G_m(x)$误分类样本的权值之和。需要注意的是，因为这里是一个弱分类器，所以e_m会满足限制条件$e_m \le \frac{1}{2}$，因为根据“弱可学习”概念，学习的准确率仅比随机猜测好，一般我们认为随机猜测的概率是$\frac{1}{2}$。

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

（c）计算$G_m(x)$的系数α_m，α_m表示分类器$G_m(x)$在最终分类器中的重要性。log为自然对数。当$e_m \le \frac{1}{2}$时，$\alpha_m \ge 0$，并且α_m随着e_m的减小而增大。所以，分类误差率越小的基本分类器在最终分类器中的作用越大。

$$\alpha_m=\frac{1}{2}\log\frac{1-e_m}{e_m}$$

（d）更新训练数据集的权值分布：

$$D_{m+1}=\left(w_{m+1,1},\ldots,w_{m+1,i},\ldots w_{m+1,N}\right)$$

$$w_{m+1,i}=\frac{w_{m,i}}{Z_m}\exp(-\alpha_m y_i G_m(x_i)),\mathrm{i}=1,2,\ldots,\mathrm{N}$$

$$Z_m=\sum_{i=1}^{N} w_{m,i}\exp(-\alpha_m y_i G_m(x_i))\ \text{规范化因子}$$

更新权值的公式可以写为：

$$w_{m+1,i}=\begin{cases}\dfrac{w_{m,i}}{Z_m}\exp(-\alpha_m),\ G_m(x_i)=y_i\\ \dfrac{w_{m,i}}{Z_m}\exp(\alpha_m),\ G_m(x_i)\neq y_i\end{cases}$$

由上式可知，被基本分类器$G_m(x)$误分类样本的权值得以扩大（$\exp(\alpha_m)\ge 1$），而被正确分类样本的权值得以缩小（$\exp(-\alpha_m<1)$）。误分类样本的权值被放大：$e^{2\alpha_m}=\frac{1-e_m}{e_m}$。

（3）构建基本分类器的线性组合：

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

最终分类器如下，$f(x)$的符号决定实例x的类别，$f(x)$的绝对值代表可信度。需要注意，α_m之和不为1。

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

四、AdaBoost算法误差分析

下面来证明AdaBoost算法是有训练误差界的。

AdaBoost算法最终分类器的训练误差界为：$\frac{1}{N}\sum_{i=1}^{N} I(\mathrm{G}(x_i) \neq y_i) \leq \frac{1}{N}\sum_i \exp\big(-y_i f(x_i)\big) = \prod_m Z_m$。所以我们需要证明分类误差是有上界的，这样我们通过上界极小化，来使得训练误差下降。

下面来一步步证明这个不等式：

（1）**首先证明左边的不等式部分：**

当$\mathrm{G}(x_i) \neq y_i$时，$y_i f(x_i) < 0$，$-y_i f(x_i) > 0$，因此$\exp\big(-y_i f(x_i)\big) > 1$，所以在$\mathrm{G}(x_i) \neq y_i$时，$I(\mathrm{G}(x_i) \neq y_i)$=1，而$\exp\big(-y_i f(x_i)\big) > 1$。

当$\mathrm{G}(x_i) = y_i$时，$y_i f(x_i) > 0$，$-y_i f(x_i) < 0$，因此$0 < \exp\big(-y_i f(x_i)\big) < 1$，所以在$\mathrm{G}(x_i) = y_i$时，$I(\mathrm{G}(x_i) \neq y_i)$=0，而$0 < \exp\big(-y_i f(x_i)\big) < 1$。

综上可得：$\frac{1}{N}\sum_{i=1}^{N} I(\mathrm{G}(x_i) \neq y_i) \leq \frac{1}{N}\sum_i \exp\big(-y_i f(x_i)\big)$。

（2）**之后来证明右边的等式部分：**

$$\frac{1}{N}\sum_i \exp\big(-y_i f(x_i)\big)$$

$$= \frac{1}{N}\sum_i \exp\left(-\sum_{m=1}^{M} \alpha_m y_i G_m(x_i)\right) \text{根据 } \mathrm{f(x)} = \sum_{\mathrm{m=1}}^{\mathrm{M}} \alpha_\mathrm{m} \mathrm{G_m(x)} \text{得}$$

$$= \frac{1}{N}\sum_i \prod_{m=1}^{M} \exp(-\alpha_m y_i G_m(x_i)) \text{根据} e^{m+n} = e^m \cdot e^n \text{ 得}$$

（根据 AdaBoost 方法，最开始初始化为均匀分布D_1）

$$= \sum_i w_{1i} \prod_{m=1}^{M} \exp(-\alpha_m y_i G_m(x_i)) \text{此时的} w_{1i} \text{就是} \frac{1}{N}$$

（根据$w_{mi}\exp(-\alpha_m y_i G_m(x_i) = Z_m w_{m+1,i}$）

$$= Z_1 \sum_i w_{2i} \prod_{m=2}^{M} \exp(-\alpha_m y_i G_m(x_i)) \text{提出} Z_1$$

$$\left(\text{重复执行提出步骤}\right)$$

$$= \prod_m Z_m$$

综上，证明完毕。

对于二分类问题的AdaBoost训练误差界，有：

$$\prod_m Z_m = \prod_{m=1}^{M} [2\sqrt{e_m(1-e_m)}] = \prod_{m=1}^{M} \sqrt{1-4\gamma_m^2} \leq \exp(-2\sum_{m=1}^{M} \gamma_m^2)$$

$$\gamma_m = \frac{1}{2} - e_m$$

证明这个不等式如下：

$$Z_m=\sum\nolimits_{i=1}^{N} w_{m,i}\exp(-\alpha_m y_i G_m(x_i))\text{如定义所示}$$

$$=\sum_{G_m(x_i)=y_i} w_{m,i}\, e^{-\alpha_m}+\sum_{G_m(x_i)\neq y_i} w_{m,i}\, e^{\alpha_m}\text{(相等的话同号为 1，不相等异号为 }-1)$$

$$=(1-e_m)e^{-\alpha_m}+e_m e^{\alpha_m}\text{根据}\mathrm{e_m}=\sum_{G_m(x_i)\neq y_i} w_{mi}\text{ 得}$$

$$=2\sqrt{e_m(1-e_m)}\text{根据}\alpha_{\mathrm{m}}=\frac{1}{2}\log\frac{1-\mathrm{e_m}}{\mathrm{e_m}}\text{得}$$

$$=\sqrt{1-4\gamma_m^2}\text{其中}\gamma_{\mathrm{m}}=\frac{1}{2}-\mathrm{e_m}$$

到这里可得到$\prod_m Z_m=\prod_{m=1}^{M}[2\sqrt{e_m(1-e_m)}]=\prod_{m=1}^{M}\sqrt{1-4\gamma_m^2}$，接下来右边的不等式证明先由$e^x$和$\sqrt{1-x}$在点$\mathrm{x}=0$的泰勒展开式推出不等式$\sqrt{1-4\gamma_m^2}\leq\exp(-2\gamma_m^2)$，进而得到。

根据$\frac{1}{N}\sum_{i=1}^{N} I(\mathrm{G}(x_i)\neq y_i)\leq\prod_m Z_m$以及$\prod_m Z_m\leq\exp(-2\sum_{m=1}^{M}\gamma_m^2)$可以得到推论，如果存在$\gamma>0$，对所有的$m$有$\gamma_m\geq\gamma$，则：

$$\frac{1}{N}\sum_{i=1}^{N} I(\mathrm{G}(x_i)\neq y_i)\leq\exp(-2M\gamma^2)$$

这个式子说明了，在此条件下，AdaBoost的训练误差是以指数速率下降的。

五、AdaBoost算法解释

Adaboost算法可以认为是：模型为加法模型，损失函数为指数函数，学习算法为前向分步算法的二类分类学习方法。

下面分别介绍加法模型、前向分步算法以及AdaBoost如何与这些联系起来。

加法模型（*additive model*）为：

$$f(x)=\sum_{m=1}^{M}\beta_m b(x;\gamma_m)$$

$b(x;\gamma_m)$为基函数，γ_m为基函数的参数，β_m为基函数的系数。$f(x)$由多个基函数加权得到，因此被称为加法模型，可以看出这个形式和AdaBoost很相似。

在给定训练数据$T=\{(x_1,y_1),(x_2,y_2),\ldots,(x_N,y_N)\}$和损失函数$L(y,f(x))$的条件下，学习加法模型$f(x)$成为损失函数极小化问题。即为：

$$\min_{\beta_m,\gamma_m}\sum_{i=1}^{N} L(y_i,\sum_{m=1}^{M}\beta_m b(x_i;\gamma_m))$$

想要针对每一个基函数，找到β_m,γ_m，使得损失函数最小。由于涉及到了多个基函数，所以这是一个复杂的问题。

前向分步算法（$forward\ stagewise\ algorithm$）的优化思想是：从前向后，每一步只学习一个基函数及其系数，逐步逼近优化目标函数式。这样每一步的优化变为：

$$\min_{\beta,\gamma}\sum_{i=1}^{N}L(y_i,\beta b(x_i;\gamma))$$

前向分步算法具体过程如下：

输入：训练数据集$T=\{(x_1,y_1),(x_2,y_2),\dots,(x_N,y_N)\}$；损失函数为$L(y,f(x))$；基函数集$\{b(x;\gamma)\}$；

输出：加法模型$f(x)$

（1）初始化$f_0(x)=0$

（2）对于$m=1,2,\dots,M$

（a）极小化损失函数，其中$\beta b(x_i;\gamma)$是我们当前处理的基函数，$f_{m-1}(x)$为前向算法之前得到的不完全加法模型。

$$(\beta_m,\gamma_m)=\arg\min_{\beta,\gamma}\sum_{i=1}^{N}L(y_i,f_{m-1}(x_i)+\beta b(x_i;\gamma))$$

（b）根据得到的(β_m,γ_m)，更新：

$$f_m(x)=f_{m-1}(x)+\beta_m(x;\gamma_m)$$

（3） 得到加法模型：

$$f(x)=f_M(x)=\sum_{m=1}^{M}\beta_m b(x;\gamma_m)$$

这样就将本来的同时求解参数转化为了逐次求解。

那么直觉上来看，加法模型和AdaBoost形式非常相似，那么如何将加法模型、前向分步算法、AdaBoost算法联系起来呢？

结论是：AdaBoost是前向分步加法算法的特例。同时，模型是基本分类器模型，损失函数是指数函数。

下面我们来证明这一结论：

（1）首先我们选取基函数为基本分类模型，这样可以将基函数$b(x;\gamma_m)$实例化为$G_m(x)$，同时基函数前面的系数β_m可以等价为AdaBoost算法中的权重α_m。这样可以得到AdaBoost加法模型的形式为：

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

（2）接下来需要证明，当AdaBoost的损失函数为指数损失函数$L\big(y,f(x)\big)=\exp[-yf(x)]$时，前向分步算法的形式和AdaBoost学习过程是一致的。

（a）假设经过m − 1轮迭代，前向分步算法已经得到$f_{m-1}(x)$：

$$f_{m-1}(x) = \alpha_1 G_1(x) + \cdots + \alpha_{m-1} G_{m-1}(x)$$

（b）继续进行第m轮迭代，得到α_m，$G_m(x)$和$f_m(x)$：

$$f_m(x) = f_{m-1}(x) + \alpha_m G_m(x)$$

（c）根据前向分步算法的定义$(\beta_m, \gamma_m) = \arg\min_{\beta,\gamma} \sum_{i=1}^{N} L(y_i, f_{m-1}(x_i) + \beta b(x_i;\gamma))$，我们需要使得根据$\alpha_m$，$G_m(x)$得到的$f_m(x)$在训练数据集上的指数损失最小：

$$\big(\alpha_m, G_m(x)\big) = \arg\min_{\alpha,G} \sum_{i=1}^{N} \exp[-y_i(f_{m-1}(x_i) + \alpha \mathrm{G}(x_i))]$$

$$= \arg\min_{\alpha,G} \sum_{i=1}^{N} \exp(-y_i f_{m-1}(x_i)) \cdot \exp(-y_i \alpha \mathrm{G}(x_i))$$

$$= \arg\min_{\alpha,G} \sum_{i=1}^{N} \overline{w_{mi}} \exp(-y_i \alpha \mathrm{G}(x_i))$$ 其中$\overline{w_{mi}} = \exp(-y_i f_{m-1}(x_i))$，与$\alpha, G$无关

（d）接下来我们说明根据前向分步算法得到的α_m^*与$G_m^*(x)$就是AdaBoost中的$\alpha_m, G_m(x)$。在AdaBoost中，α_m为：

$$\alpha_m = \frac{1}{2}\log\frac{1-e_m}{e_m}$$

（e）我们先修改一下上面的式子便于理解：

我们想要找到一个G使得下式最小，此时α可以看做固定的：

$$\min \sum_{i=1}^{N} \overline{w_{mi}} \exp(-y_i \alpha \mathrm{G}(x_i))$$

由于$\overline{w_{mi}}$与α, G无关，所以修改为：

$$\min \sum_{i=1}^{N} \exp(-y_i \alpha \mathrm{G}(x_i))$$

根据y_i与$\mathrm{G}(x_i)$的关系，继续修改：

$$\min \Big(\sum_{G_m(x_i)=y_i} e^{-\alpha} + \sum_{G_m(x_i)\neq y_i} e^{\alpha} \Big)$$

由于$\alpha > 0$，所以我们应该尽可能增加$G_m(x_i) = y_i$的情况，减少$G_m(x_i) \neq y_i$的情况。这样就等价于下式，相当于使得加权数据分类误差率最小。

$$G_m^*(x) = \arg\min_{G} \sum_{i=1}^{N} \overline{w_{mi}} I(G_m(x_i) \neq y_i)$$

因此，$G_m^*(x)$即为AdaBoost算法的基本分类器$G_m(x)$。

（f）接下来证明α_m^*：

$$\sum_{i=1}^{N}\overline{w_{mi}}\exp\left(-y_i\alpha \mathrm{G}(x_i)\right)$$
$$=\sum_{G_m(x_i)=y_i}\overline{w_{mi}}e^{-\alpha}+\sum_{G_m(x_i)\neq y_i}\overline{w_{mi}}e^{\alpha}$$
$$=\sum_{G_m(x_i)=y_i}\overline{w_{mi}}e^{-\alpha}(1-I(G_m(x_i)\neq y_i))+\sum_{G_m(x_i)\neq y_i}\overline{w_{mi}}e^{\alpha}I(G_m(x_i)\neq y_i)$$
$$=(e^{\alpha}-e^{-\alpha})\sum_{i=1}^{N}\overline{w_{mi}}I(G_m(x_i)\neq y_i)+e^{-\alpha}\sum_{i=1}^{N}\overline{w_{mi}}$$

将$G_m^*(x)$代入上式可得，对α求导使导数为 0，得：

$$\alpha_m^*=\frac{1}{2}\log\frac{1-e_m}{e_m}$$

发现与AdaBoost中的形式一致。

（g）最后来看每一轮的样本权值更新。由下式：

$$f_m(x)=f_{m-1}(x)+\alpha_m G_m(x)$$
$$\overline{w_{mi}}=\exp(-y_i f_{m-1}(x_i))$$

可得：

$$\overline{w_{m+1,i}}=\overline{w_{mi}}\exp(-y_i\alpha_m G_m(x))$$

与AdaBoost权值更新公式$w_{m+1,i}=\frac{w_{m,i}}{Z_m}\exp(-\alpha_m y_i G_m(x_i))$相比，只差规范化因子$Z_m$，因此等价。

六、回顾：决策树

分类与回归树（classification and regression tree，CART）是应用广泛的决策树学习方法。由特征选择、树的生成以及剪枝组成。既可以用于分类也可以用于回归。CART是在给定输入随机变量X条件下输出随机变量Y的条件概率分布的学习方法。CART假设决策树是二叉树，内部节点特征的取值为“是”和“否”。

决策树的生成就是递归地构建二叉决策树的过程，对回归树用平方误差最小化准则，对分类树用基尼指数最小化准则，进行特征选择，生成二叉树。

下面回顾一下CART回归树的生成。

假设X与Y分别为输入和输出变量，并且Y是连续变量，给定训练数据集：$D=\{(x_1,y_1),(x_2,y_2),\dots,(x_N,y_N)\}$。一个回归树对应着输入空间（特征空间）的一个划分以及在划分的单元上的输出值。假设输入空间划分为$R_1,R_2,\dots,R_M$，在每个划分上有对应的输出值c_m，于是回归树模型可以表示为：$f(x)=\sum_{m=1}^{M}c_m I_m(x\in R_m)$。

我们想要找到一个对于输入空间的划分，使得平方误差最小：$\sum_{x_i \in R_m}(y_i - f(x_i))^2$。采用启发式规则，选择第j个变量$x^{(j)}$和他的取值s。作为切分点，定义两个区域：

$$R_1(j,s) = \{x|x^{(j)} \le s\}$$
$$R_2(j,s) = \{x|x^{(j)} > s\}$$

求解过程如下：

$$\hat{c_1} = ave(y_i|x_i \in R_1(j,s))$$
$$\hat{c_2} = ave(y_i|x_i \in R_2(j,s))$$
$$\min_{j,s}[\min_{c_1} \sum_{x_i \in R_1(j,s)} (y_i - c_1)^2 + \min_{c_2} \sum_{x_i \in R_2(j,s)} (y_i - c_2)^2]$$

遍历所有输入变量，找到最优的切分变量j。划分为两个区域以后重复这个过程，这样的回归树通常被称为最小二乘回归树。

七、提升树

提升树是以分类树或回归树为基本分类器的提升方法。以决策树为基函数的提升方法称为提升树（boosting tree），对分类问题决策树是二叉分类树，对回归问题决策树是二叉回归树。

提升树模型可以表示为决策树的加法模型：

$$f_M(x) = \sum_{m=1}^{M} T(x;\Theta_m)$$

其中，$T(x;\Theta_m)$表示决策树，Θ_m为决策树参数，M为树的个数。

提升树算法采用前向分步算法，首先确定初始提升树$f_0(x) = 0$，第m步的模型是：$f_m(x) = f_{m-1}(x) + T(x;\Theta_m)$，通过经验损失极小化来确定$\Theta_m$，$\widehat{\Theta_m} = \arg\min_{\Theta_m} \sum_{i=1}^{N} L(y_i, f_{m-1}(x_i) + T(x_i;\Theta_m))$。

针对不同问题的提升树学习算法，主要区别在于使用的损失函数不同，包括用平方误差损失函数的回归问题，用指数损失函数的分类问题，以及用一般损失函数的一般决策问题。

下面主要叙述回归问题的提升树。

已知一个训练数据集$T = \{(x_1,y_1),(x_2,y_2),\dots,(x_N,y_N)\}$，$x_i \in \mathcal{X} \subseteq R^n$，$y_i \in \mathcal{Y} \subseteq R$，如果将输入空间$\mathcal{X}$划分为$J$个互不相交的区域$R_1,R_2,\dots,R_J$，且在每个区域上输出常量$c_j$，则树可以表示为：

$$T(x;\Theta) = \sum_{j=1}^{J} c_j I(x \in R_j)$$

如果采用平方误差损失函数：

$$L(y,f(x)) = (y-f(x))^2$$

损失为：

$$\begin{aligned} &L(y, f_{m-1}(x) + T(x;\Theta_m)) \\ &= [y - f_{m-1}(x) - T(x;\Theta_m)]^2 \\ &= [r - T(x;\Theta_m)]^2 \end{aligned}$$

此时：$r = y - f_{m-1}(x)$，是当前模型拟合数据的残差，所以，回归问题的提升树算法来说，只需简单地拟合模型的残差，也就是说使得$T(x;\Theta_m)$尽可能靠近r。

所以，回归问题的提升树算法为：

输入：$T = \{(x_1,y_1),(x_2,y_2),\dots,(x_N,y_N)\}$，$x_i \in \mathcal{X} \subseteq R^n$，$y_i \in \mathcal{Y} \subseteq R$

输出：提升树$f_M(x)$

（1） 初始化$f_0(x)$

（2） 对$m = 1,2,\dots,M$

a. 计算残差$r_{mi} = y_i - f_{m-1}(x_i)$，$i = 1,2,\dots,N$

b. 模拟残差r_{mi}学习回归树，得到$T(x;\Theta_m)$

c. 更新$f_m(x) = f_{m-1}(x) + T(x;\Theta_m)$

（3） 得到回归问题提升树：$f_M(x) = \sum_{m=1}^{M} T(x;\Theta_m)$

对于平方损失和指数损失，我们可以经过一定的推导，得到学习方法的简化形式，但是对于更一般化的损失函数，优化并不是那么容易。因此有了梯度提升（gradient boosting）方法，非常类似我们平时用的梯度下降方法。核心思想是利用损失函数负梯度在当前模型的值$-[\frac{\partial L(y_i,f(x_i))}{\partial f(x_i)}]_{f(x)=f_{m-1}(x)}$作为回归问题提升树算法中的残差的近似值，拟合一个回归树。

八、参考

- 机器学习实战
- 统计学习方法
- http://www.hankcs.com/ml/adaboost.html
- https://blog.csdn.net/xueyingxue001/article/details/51304430
- https://blog.csdn.net/haidao2009/article/details/7514787
- http://www.hankcs.com/ml/adaboost.html
- https://zh.wikipedia.org/wiki/%E6%8F%90%E5%8D%87%E6%96%B9%E6%B3%95