机器学习算法之决策树

1、什么是决策树

决策树是一个预测模型，他代表的是对象属性与对象值之间的一种映射关系。树中每个非叶子结点表示某个对象，而每个分叉路径则代表的某个可能的属性值，而每个叶结点则对应从根节点到该叶节点所经历的路径所表示的对象的值，这是一种依托于分类、训练上的预测树，从训练数据集中归纳出一组分类规则。

2、决策树学习

决策树学习算法通常是一个递归地选择最优特征，并根据该特征对训练数据进行分割，使得对各个子数据集有一个最好的分类的过程。这一过程对应着对特征空间的划分，也对应着决策树的构建。决策树的学习通常包含三个步骤：特征选择，决策树的生成和决策树的修剪。

3、ID3 算法

ID3 算法的核心思想就是以信息增益度量属性选择，选择分裂后信息增益最大的属性进行分裂。该算法采用自顶向下的贪婪搜索遍历可能的决策树空间。

3.1、 特征选择

ID3 算法的核心问题是选取在树的每个结点要测试的属性。我们希望选择的是最有利于分类实例的属性，信息增益是用来衡量给定的属性区分训练样例的能力，而 ID3 算法在增长树的每一步使用信息增益从候选属性中选择属性。

而信息增益的度量标准是熵，所以下面先给出熵的定义：熵是信息论中的概念，它刻画了任意样例集的纯度，当数据变得越来越纯净时，熵值变得越来越小。

设 D 为用类别对训练元组进行的划分，则 D 的熵（entropy）表示为：

$$\text{entropy}(D) = -\sum_{i=1}^{m} p_i \log_2(p_i)$$

其中 pi 表示第 i 个类别在整个训练元组中出现的概率。

举例来说，假设 S 是一个关于布尔概念的有 14 个样例的集合，它包括 9 个正例和 5 个反例（我们采用记号[9+，5-]来概括这样的数据样例），那么 S 相对于这个布尔样例的熵为：

Entropy（[9+，5-]）=-（9/14）log2（9/14）-（5/14）log2（5/14）=0.94

现在我们假设将训练元组 D 按属性 A 进行划分，则可以将 D 划分成 v 个不相交的子集 D_1，D_2，…，D_v，划分后 D 的熵为：

$$\text{entropy}_{A_i}(D) = -\sum_{j=1}^{v} \frac{D_j}{D} entropy(D_j)$$

简单的说，一个属性的信息增益就是由于使用这个属性分割样例而导致的期望熵降低(或者说，样本按照某属性划分时造成熵减少的期望)：

$$\text{gain}(D, A_i) = entropy(D) - entropy_{A_i}(D)$$

根据信息增益准则的特征选择方法是：对训练数据集（或子集）D，计算其每个特征的信息增益，并比较它们的大小，选择信息增益最大的特征。

下表是一个由 14 个样本组成的气象数据，并已知这些天气下是否打球，数据包括天气、温度、湿度和是否有风 4 个属性，并用 P 表示去打球，N 表示没有去打球。

id	outlook	temperature	humidity	windy	class
1	sunny	hot	high	false	N
2	sunny	hot	high	true	N
3	overcast	hot	high	false	P
4	rain	mild	high	false	P
5	rain	cool	normal	false	P
6	rain	cool	normal	true	N
7	overcast	cool	normal	true	P
8	sunny	mild	high	false	N
9	sunny	cool	normal	false	P
10	rain	mild	normal	false	P
11	sunny	mild	normal	true	P
12	overcast	mild	high	true	P
13	overcast	hot	normal	false	P
14	rain	mild	high	true	N

我们根据以上数据集 D 计算 4 个属性的信息增益，首先计算 D 的熵：

$$-\frac{9}{14}\log_2\frac{9}{14}-\frac{5}{14}\log_2\frac{5}{14}=0.940$$

然后计算各特征对数据集 D 的信息增益，对每项指标分别统计：在不同的取值下打球和不打球的次数。

outlook			**temperature**			**humidity**			**windy**			**play**	
	yes	no		yes	no		yes	no		yes	no	P	N
sunny	2	3	hot	2	2	high	3	4	false	6	2	9	5
overcast	4	0	mild	4	2	normal	6	1	true	3	3		
rainy	3	2	cool	3	1								

我们先划分 outlook 属性的数据，可分为三类，sunny、overcast、rainy，则划分后的熵为：

$$entropy_{outlook}=\frac{5}{14}\times\left(-\frac{2}{5}\log_2\frac{2}{5}-\frac{3}{5}\log_2\frac{3}{5}\right)+\frac{4}{14}\times\left(-1\log_2 1-0\log_2 0\right)+\frac{5}{14}\times\left(-\frac{3}{5}\log_2\frac{3}{5}-\frac{2}{5}\log_2\frac{2}{5}\right)$$

$$entropy_{outlook}=\frac{5}{14}\times 0.971+\frac{4}{14}\times 0+\frac{5}{14}\times 0.971=0.693$$

信息增益为：

$$\text{gain}(D,outlook)=0.94-0.693=0.247$$

类似地，我们用 temperature 属性将数据划分为 hot、middle、cool 三类。

$$entropy_{temperature}=\frac{4}{14}\times\left(-\frac{1}{2}\log_2\frac{1}{2}-\frac{1}{2}\log_2\frac{1}{2}\right)+\frac{6}{14}\times\left(-\frac{2}{3}\log_2\frac{2}{3}-\frac{1}{3}\log_2\frac{2}{3}\right)+\frac{4}{14}\times\left(-\frac{3}{4}\log_2\frac{3}{4}-\frac{1}{4}\log_2\frac{1}{4}\right)$$

$$=0.911$$

$$\text{gain}(D,temperature)=0.94-0.911=0.029$$

以此类推，可以计算得到：

$$\text{gain}(D,\text{humidity})=0.152$$

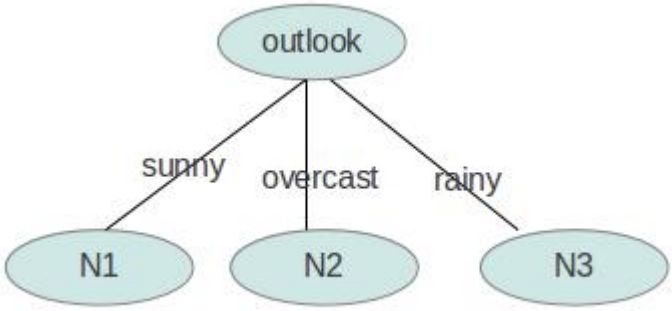
$$\text{gain}(D,\text{windy})=0.048$$

比较各信息增益，可知 outlook 属性的信息增益最大，所以选择特征 outlook 作为最优特征。

3.2、ID3 算法决策树生成

ID3 算法的核心是在决策树各个节点上应用信息增益准则选择特征，递归地构建决策树。

根据上述计算过程，我们可以把 outlook 作为根节点，它将数据集划分为 3 个子集，如下图。接下来要确定 N1 的属性，再计算划分后的 outlook=sunny 数据集中的各个特征的信息增益，选择信息增益值最大的属性作为 N1 处的节点，以此类推构造决策树，直到所有特征的信息增益均小于一个阈值或没有特征可以选择为止。



3.3、算法优缺点

优点：

1）分类精度高

2）模型简单

3）对噪声数据有很好的健壮性

缺点：

1）只能处理离散数据

2）偏向于优先选择有较多可能取值的属性

3）生成的树容易产生过拟合

4、C4.5 算法

C4.5 算法为 ID3 算法的改进，它是决策树的核心算法，相对于 ID3 算法改进的地方为：

1）用信息增益率来选择属性。ID3 使用的是信息增益而 C4.5 用的是信息增

益率。对，区别就在于一个是信息增益，一个是信息增益率。可以校正以信息增益选择特征时存在偏向于选择取值较多的特征的问题。

2）在树构造过程中进行剪枝，在构造决策树的时候，那些挂着几个元素的节点，不考虑最好，不然容易导致过拟合。

3）对非离散数据也能处理。

4）能够对不完整数据进行处理

4.1、特征选择

既然 C4.5 算法是使用信息增益率进行特征选择，那么接下来给出信息增益率的定义。

增益比率度量是用前面的信息增益 $\text{gain}(D,A_i)$ 和分裂信息 $splitInfo(D,A_i)$ 来共同定义的：

$$\text{gain}Ratio(D,A_i)=\frac{gain(D,A_i)}{splitInfo(D,A_i)}$$

其中：$\text{split}Info(D,A_i)=-\sum_{j=1}^{s}\left(\frac{|\text{D}_\text{j}|}{\text{D}}\times\log_2\frac{|\text{D}_\text{j}|}{\text{D}}\right)$。

S 是属性 A_i 的可能取值的数目， D_j 是数据集中具有 A_i 属性的第 j 个值的子集。

根据 3.1 节所示的例子，我们已经计算出了各属性的信息增益，接下来我们再计算一下各属性的信息增益率。

因为信息增益在上面已经计算过，所以接下来需要计算分裂信息，对于属性 outlook，它有 3 个值 sunny、rainy 和 overcast，则

$$splitInfo(D,outlook)=-\frac{5}{14}\times\log_2(\frac{5}{14})-\frac{5}{14}\times\log_2(\frac{5}{14})-\frac{4}{14}\times\log_2(\frac{4}{14})=1.577$$

同样的，对于属性 temperature、humidity 和 windy 的分裂信息分别为 1.557、1.0 和 0.985，所以他们的信息增益率为：

$$gainRatio(D,outlook)=0.247/1.577=0.156$$

$$gainRatio(D,temperature)=0.029/1.557=0.019$$

$$gainRatio(D,humidity)=0.152/1.0=0.152$$

$$gainRatio(D,windy)=0.048/0.985=0.049$$

最终选择 outlook 作为最优特征。

4.2、算法优缺点

优点：

1）产生的分类规则易于理解

2）准确率较高

缺点：

1）在构造树的过程中，需要对数据集进行多次的顺序扫描和排序，因而导

致算法的低效

2）只适合于能够驻留于内存的数据集，当训练集大得无法在内存容纳时程序无法进行

5 、CART 之分类树

分类与回归树（classification and regression tree，CART）模型由 Breiman 等人在 1984 年提出，是应用广泛的决策树学习方法．CART 同样由特征选择、树的生成及剪枝组成，既可以用于分类也可以用于回归．以下将用于分类与回归的树统称为决策树．

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，内部结点特征的取值为“是”和“否”，左分支是取值为“是”的分支，右分支是取值为“否”的分支．这样的决策树等价于递归地二分每个特征，将输入空间即特征空间划分为有限个单元，并在这些单元上确定预测的概率分布，也就是在输入给定的条件下输出的条件概率分布．

定义 5.4（基尼指数） 分类问题中，假设有 K 个类，样本点属于第 k 类的概率为 p_k，则概率分布的基尼指数定义为

$$\text{Gini}(p)=\sum_{k=1}^{K}p_k(1-p_k)=1-\sum_{k=1}^{K}p_k^2 \tag{5.22}$$

如果样本集合 D 根据特征 A 是否取某一可能值 a 被分割成 D_1 和 D_2 两部分，即

$$D_1=\{(x,y)\in D\mid A(x)=a\},\quad D_2=D-D_1$$

则在特征 A 的条件下，集合 D 的基尼指数定义为

$$\text{Gini}(D,A)=\frac{|D_1|}{|D|}\text{Gini}(D_1)+\frac{|D_2|}{|D|}\text{Gini}(D_2) \tag{5.25}$$

基尼指数 $\text{Gini}(D)$ 表示集合 D 的不确定性，基尼指数 $\text{Gini}(D,A)$ 表示经 $A=a$ 分割后集合 D 的不确定性．基尼指数值越大，样本集合的不确定性也就越大，这一点与熵相似．

6、决策树剪枝

决策树生成算法递归地产生决策树，直到不能继续下去为止。这样产生的树往往对训练数据的分类很准确，但对未知的测试数据的分类却没有那么准确，即出现过拟合现象。过拟合的原因在于学习时过多的考虑如何提高对训练数据的正确分类，从而构建过于复杂的决策树。

为了减少过拟合问题，我们需要对树进行剪枝，即删除一些子树或分支，用多个类的叶节点代替这些分支。有两种常用的决策树剪枝处理方法，即预剪枝和后剪枝。

5.1 预剪枝

在预剪枝方法中，最直接的方法是通过提前设置决策树生长的最大深度，使决策树不能充分生长，进而达到预剪枝目的。预剪枝的核心问题是如何事先指定

树的最大深度，如果设置的最大深度不恰当，那么将会导致过于限制树的生长，使决策树的表达式规则趋于一般，不能更好地对新数据集进行分类和预测。

除了事先限定决策树的最大深度之外，还有另外一个方法来实现预剪枝操作，那就是采用检验技术对当前结点对应的样本集合进行检验，如果该样本集合的样本数量已小于事先指定的最小允许值，那么停止该结点的继续生长，并将该结点变为叶子结点，否则可以继续扩展该结点。

预剪枝另一方法是在决策树的生成过程中，对每个结点在划分前进行估计，若当前结点的划分不能带来决策树泛化性能的提升，则停止划分并将当前节点标记为叶结点。

5.2 后剪枝

后剪枝则是从训练集生成一颗完整的决策树，然后自底向上地对非叶结点进行考察，若将该结点对应的子树替换为叶节点能带来决策树泛化性能的提升，则将该子树替换为叶结点。因为后剪枝比预剪枝更有效，ID3、C4.5 和 CART 算法都是采用的后剪枝方法。

Reduced-Error Pruning(REP,错误率降低剪枝）

该剪枝方法一般规则标准是：将树上的每个节点作为剪枝候选对象，然后进行以下操作：

1）删除以此结点为根的子树

2）使其成为叶子结点

3）赋予该结点关联的训练数据的最常见分类

4）当修剪后的树对于验证集合的性能不会比原来的树差时，即修剪后的树的错误率比原来的错误率低时，才真正删除该结点

因为训练集合的过拟合，使得验证集合数据能够对其进行修正，反复进行上面的操作，从底向上的处理结点，删除那些能够最大限度的提高验证集合的精度的结点，直到进一步修剪有害为止(有害是指修剪会减低验证集合的精度)。

REP 是最简单的后剪枝方法之一，不过在数据量比较少的情况下，REP 方法趋于过拟合而较少使用。这是因为训练数据集合中的特性在剪枝过程中被忽略，所以在验证数据集合比训练数据集合小的多时，要注意这个问题。

尽管 REP 有这个缺点，不过 REP 仍然作为一种基准来评价其它剪枝算法的性能。它对于两阶段决策树学习方法的优点和缺点提供了了一个很好的学习思路。由于验证集合没有参与决策树的创建，所以用 REP 剪枝后的决策树对于测试样例的偏差要好很多，能够解决一定程度的过拟合问题。

Cost-Complexity Pruning(CCP、代价复杂度)（《统计学习方法》p65）

Pessimistic Error Pruning(PEP，悲观剪枝）

http://blog.sina.com.cn/s/blog_4e4dec6c0101fdz6.html

(此链接为部分剪枝方法)

6、缺失值的处理

对于某些采样数据，可能会缺少属性值。在这种情况下，处理缺少属性值的通常做法是赋予该属性的常见值，或者属性均值。另外一种比较好的方法是为该属性的每个可能值赋予一个概率，即将该属性以概率形式赋值。

7、随机森林

http://blog.csdn.net/zrjdds/article/details/50133843

随机森林在变量（列）的使用和数据（行）的使用上进行随机化，生成很多分类树，再汇总分类树的结果。

鉴于决策树容易过拟合的缺点，随机森林采用多个决策树的投票机制来改善决策树，我们假设随机森林使用了 m 棵决策树，那么就需要产生 m 个一定数量的样本集来训练每一棵树，如果用全样本去训练 m 棵决策树显然是不可取的，全样本训练忽视了局部样本的规律，对于模型的泛化能力是有害的

产生 n 个样本的方法采用 Bootstraping 法，这是一种有放回的抽样方法，产生 n 个样本

而最终结果采用 Bagging 的策略来获得，即多数投票机制

随机森林的生成方法：

1. 从样本集中通过重采样的方式产生 n 个样本
2. 假设样本特征数目为 a，对 n 个样本选择 a 中的 k 个特征，用建立决策树的方式获得最佳分割点
3. 重复 m 次，产生 m 棵决策树
4. 多数投票机制来进行预测

（需要注意的一点是，这里 m 是指循环的次数，n 是指样本的数目，n 个样本构成训练的样本集，而 m 次循环中又会产生 m 个这样的样本集）

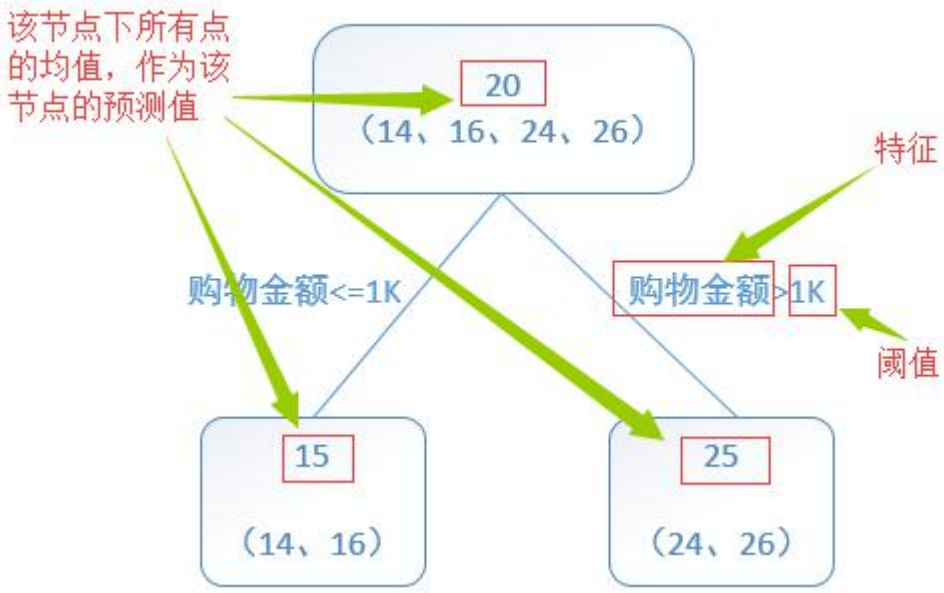
优缺点

随机森林是一个比较优秀的模型，在我的项目的使用效果上来看，它对于多维特征的数据集分类有很高的效率，还可以做特征重要性的选择。运行效率和准确率较高，实现起来也比较简单。但是在数据噪音比较大的情况下会过拟合，过拟合的缺点对于随机森林来说还是较为致命的。

回归树（《统计学习方法》p68）

http://www.jianshu.com/p/005a4e6ac775

回归树总体流程类似于分类树，区别在于，回归树的每一个节点都会得一个预测值，以年龄为例，该预测值等于属于这个节点的所有人年龄的平均值。分枝时穷举每一个 feature 的每个阈值找最好的分割点，但衡量最好的标准不再是最大熵，而是最小化平方误差。也就是被预测出错的人数越多，错的越离谱，平方误差就越大，通过最小化平方误差能够找到最可靠的分枝依据。分枝直到每个叶子节点上人的年龄都唯一或者达到预设的终止条件(如叶子个数上限)，若最终叶子节点上人的年龄不唯一，则以该节点上所有人的平均年龄做为该叶子节点的预测年龄。



算法 5.5（最小二乘回归树生成算法）

输入：训练数据集 D；

输出：回归树 $f(x)$.

在训练数据集所在的输入空间中，递归地将每个区域划分为两个子区域并决定每个子区域上的输出值，构建二叉决策树：

（1）选择最优切分变量 j 与切分点 s，求解

$$\min_{j,s}\left[\min_{c_1}\sum_{x_i\in R_1(j,s)}(y_i-c_1)^2+\min_{c_2}\sum_{x_i\in R_2(j,s)}(y_i-c_2)^2\right] \tag{5.21}$$

遍历变量 j，对固定的切分变量 j 扫描切分点 s，选择使式 (5.21) 达到最小值的对 (j,s).

（2）用选定的对 (j,s) 划分区域并决定相应的输出值：

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

$$\hat{c}_m=\frac{1}{N_m}\sum_{x_i\in R_m(j,s)}y_i,\quad x\in R_m,\quad m=1,2$$

（3）继续对两个子区域调用步骤 (1)，(2)，直至满足停止条件.

（4）将输入空间划分为 M 个区域 $R_1,R_2,\cdots,R_M$，生成决策树：

$$f(x)=\sum_{m=1}^{M}\hat{c}_m I(x\in R_m)$$

■

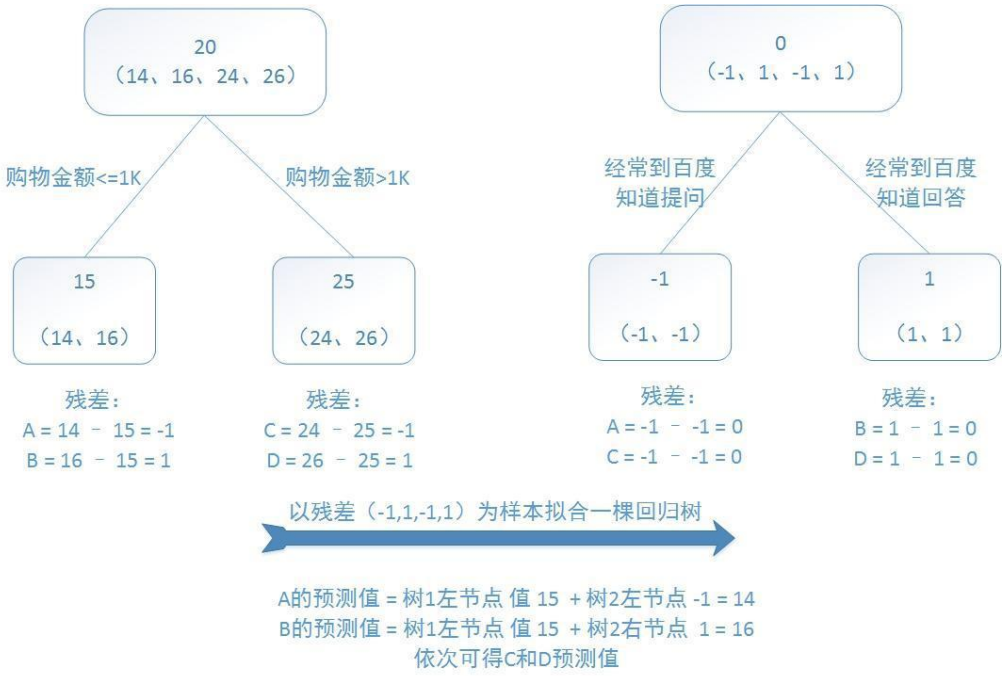
提升树（《统计学习方法》p147）

提升树是迭代多棵回归树来共同决策。当采用平方误差损失函数时，每一棵回归树学习的是之前所有树的结论和残差，拟合得到一个当前的残差回归树，残差的意义如公式：残差 = 真实值 - 预测值 。提升树即是整个迭代过程生成的回归树的累加。

举个例子，参考自一篇博客（参考文献 4），该博客举出的例子较直观地展现出多棵决策树线性求和过程以及残差的意义。

训练一个提升树模型来预测年龄：

训练集是 4 个人，A，B，C，D 年龄分别是 14，16，24，26。样本中有购物金额、上网时长、经常到百度知道提问等特征。提升树的过程如下：



因此，给定当前模型 $fm-1(x)$，只需要简单的拟合当前模型的残差。现将回归问题的提升树算法叙述如下：

（1）初始化 $f_0(x)=0$

（2）对 $m=1,2,\ldots,M$

（a）计算残差

$$r_{mi}=y_i-f_{m-1}(x_i),i=1,2,\cdots,N$$

（b）拟合残差 r_{mi} 学习一个回归树，得到 $T(x;\varnothing m)$

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

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

提升树算法

梯度提升树

提升树利用加法模型和前向分步算法实现学习的优化过程。当损失函数时平方损失和指数损失函数时，每一步的优化很简单，如平方损失函数学习残差回归树。

TABLE 10.2. *Gradients for commonly used loss functions.*

Setting	Loss Function	$-\partial L(y_i, f(x_i))/\partial f(x_i)$
Regression	$\frac{1}{2}[y_i - f(x_i)]^2$	$y_i - f(x_i)$
Regression	$\lvert y_i - f(x_i)\rvert$	$\text{sign}[y_i - f(x_i)]$
Regression	Huber	$y_i - f(x_i)$ for $\lvert y_i - f(x_i)\rvert \le \delta_m$ $\delta_m \text{sign}[y_i - f(x_i)]$ for $\lvert y_i - f(x_i)\rvert > \delta_m$ where $\delta_m = \alpha$th-quantile$\{\lvert y_i - f(x_i)\rvert\}$
Classification	Deviance	kth component: $I(y_i = \mathcal{G}_k) - p_k(x_i)$

但对于一般的损失函数，往往每一步优化没那么容易，如上图中的绝对值损失函数和 Huber 损失函数。针对这一问题，Freidman 提出了梯度提升算法：利用最速下降的近似方法，即利用损失函数的负梯度在当前模型的值，作为回归问题中提升树算法的残差的近似值，拟合一个回归树。（注：鄙人私以为，与其说负梯度作为残差的近似值，不如说残差是负梯度的一种特例）算法如下（截图来自《The Elements of Statistical Learning》）：

Algorithm 10.3 *Gradient Tree Boosting Algorithm.*

1. Initialize $f_0(x) = \arg\min_\gamma \sum_{i=1}^{N} L(y_i, \gamma)$.
2. For $m = 1$ to M:
   (a) For $i = 1, 2, \ldots, N$ compute
   $$r_{im} = -\left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f=f_{m-1}}.$$
   (b) Fit a regression tree to the targets r_{im} giving terminal regions R_{jm}, $j = 1, 2, \ldots, J_m$.
   (c) For $j = 1, 2, \ldots, J_m$ compute
   $$\gamma_{jm} = \arg\min_\gamma \sum_{x_i \in R_{jm}} L\left(y_i, f_{m-1}(x_i) + \gamma\right).$$
   (d) Update $f_m(x) = f_{m-1}(x) + \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm})$.
3. Output $\hat{f}(x) = f_M(x)$.