Gradient Boosted Decision Tree

一、Adaptive Boosted Decision Tree

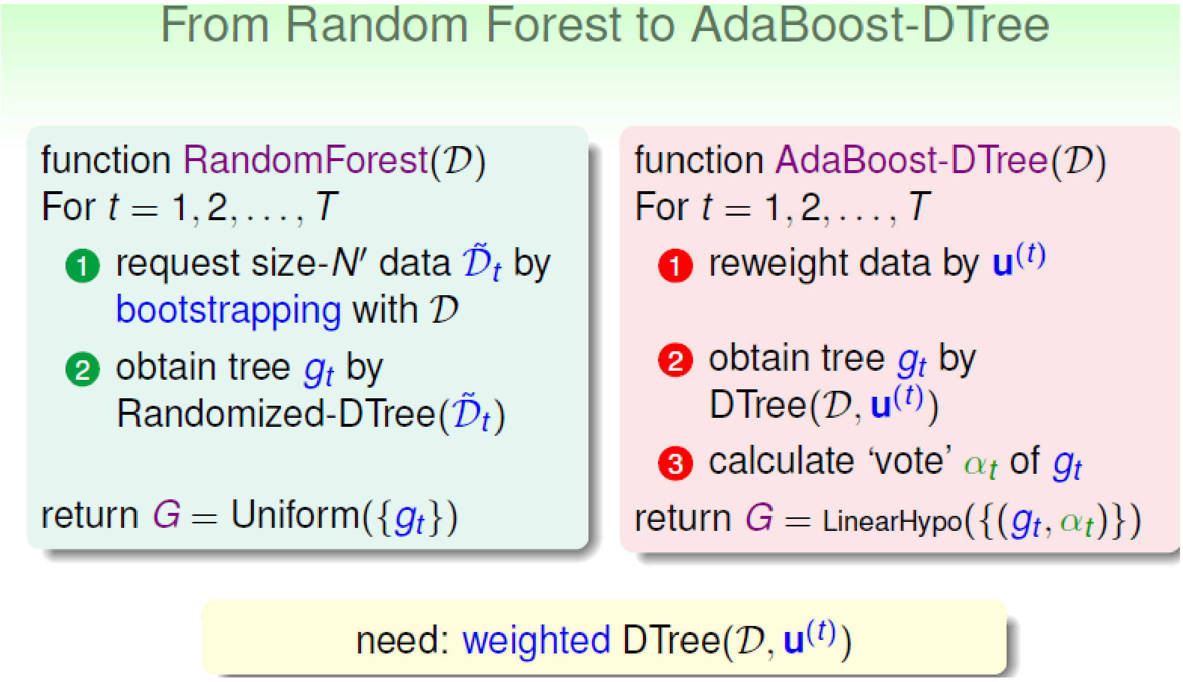
1. 对比 **Decision Tree** 和 **AdaBoost-Tree**

   我们总结过关于几类aggregation的例子

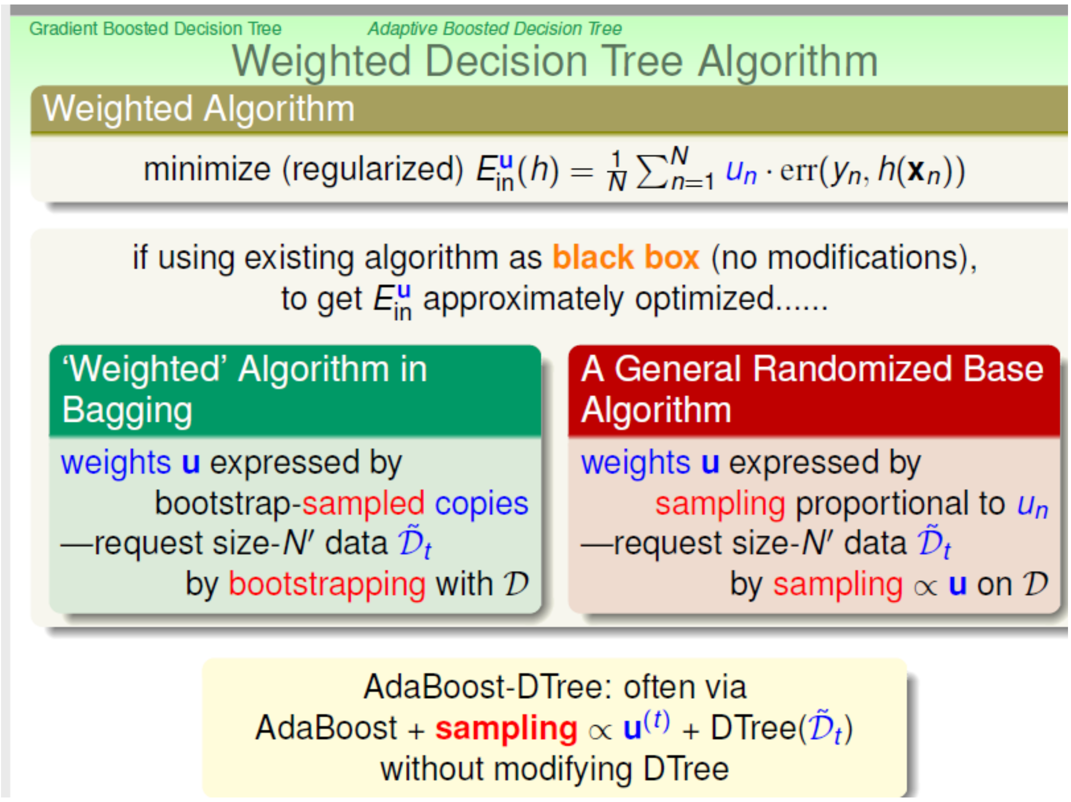
   - Bagging (boostrap) - uniform
   - AdaBoost - non-uniform
   - Decision Tree - conditionally

   uniformly的Decision Tree就是Random Forest

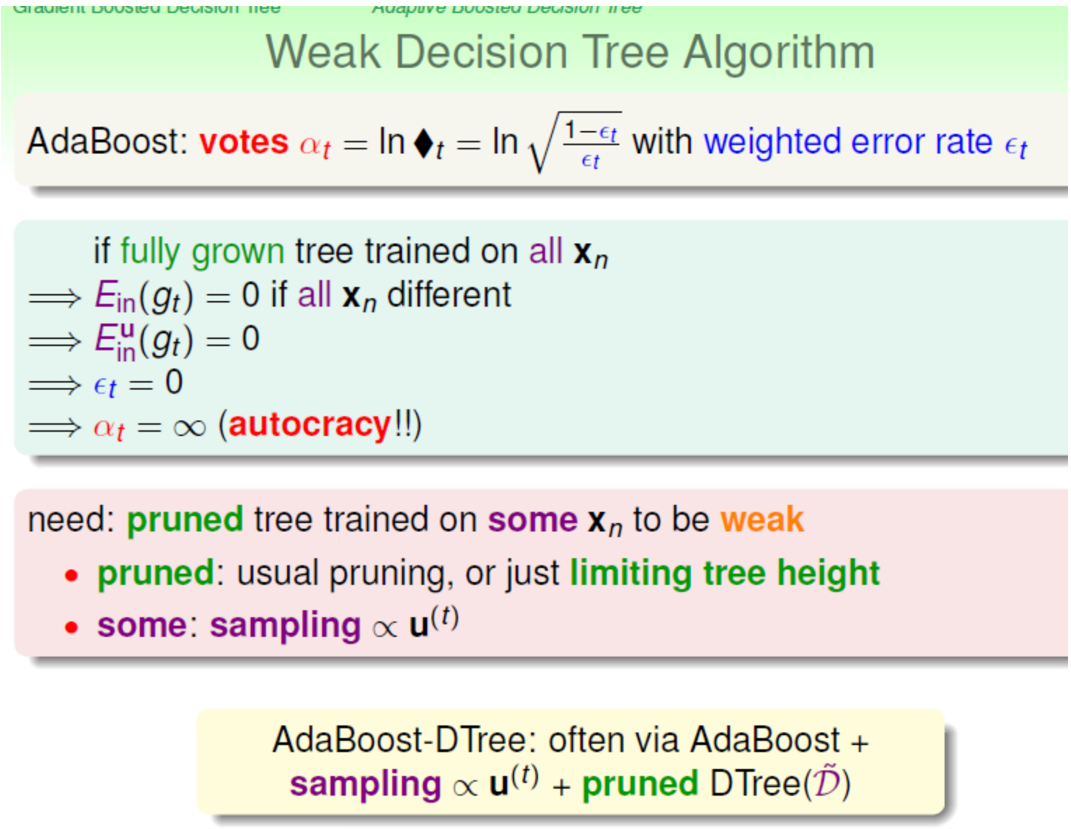
   配合weights的Decision Tree就是AdaBoost-DTree



2. 于是我们试图在Decision-Tree中加入权重，但是我们希望让DT成为一个黑盒子而不是直接对这个模型进行大幅度的改动，因此我们希望抽样的时候就按照权重的比例来抽样，从而我们不需要再对于原来的算法进行修改：



3. 我们不希望有一棵树太强，这样整个模型的效果就是去了，于是我们希望通过剪枝来改善，或者仅仅是限制树的高度。



4. 极端的剪枝条件下，这个AdaBoost-DT就是一个AdaBoost-Stump

AdaBoost with Extremely-Pruned Tree

what if DTree with **height** ≤ 1 (extremely pruned)?

DTree (C&RT) with height ≤ 1

learn branching criteria

$$b(\mathbf{x}) = \underset{\text{decision stumps } h(\mathbf{x})}{\operatorname{argmin}} \sum_{c=1}^{2} |\mathcal{D}_c \text{ with } h| \cdot \text{impurity}(\mathcal{D}_c \text{ with } h)$$

—if impurity = binary classification error,

just a decision stump, remember? :-)

AdaBoost-Stump
= **special case** of AdaBoost-DTree

二、Optimization View of AdaBoost

1. 回忆一下在AdaBoost中权重的更新方法，我们有如下的推导

Example Weights of AdaBoost

$$u_n^{(t+1)} = \begin{cases} u_n^{(t)} \cdot \blacklozenge_t & \textbf{if incorrect} \\ u_n^{(t)} / \blacklozenge_t & \textbf{if correct} \end{cases}$$

$$= u_n^{(t)} \cdot \blacklozenge_t^{-y_n g_t(\mathbf{x}_n)} = u_n^{(t)} \cdot \exp\left(-y_n \alpha_t g_t(\mathbf{x}_n)\right)$$

$$u_n^{(T+1)} = u_n^{(1)} \cdot \prod_{t=1}^{T} \exp(-y_n \alpha_t g_t(\mathbf{x}_n)) = \frac{1}{N} \cdot \exp\left(-y_n \sum_{t=1}^{T} \alpha_t g_t(\mathbf{x}_n)\right)$$

- recall: $G(\mathbf{x}) = \operatorname{sign}\left(\sum_{t=1}^{T} \alpha_t g_t(\mathbf{x})\right)$
- $\sum_{t=1}^{T} \alpha_t g_t(\mathbf{x})$: **voting score** of $\{g_t\}$ on $\mathbf{x}$

AdaBoost: $u_n^{(T+1)} \propto \exp(-y_n(\text{voting score on } \mathbf{x}_n))$

AdaBoost当中的权重和 voting score有着一些关系

2. 类比我们在SVM中得到的Margin，我们可以看出实际上voting score是一种有符号且没有被归一化的margin，进而我们需要让voting score是很大的正数就需要u_n^{T+1}足够小

结论：AdaBoost 可以减小$\sum_{n=1}^{N} u_n^t$

Gradient Boosted Decision Tree — Optimization View of AdaBoost

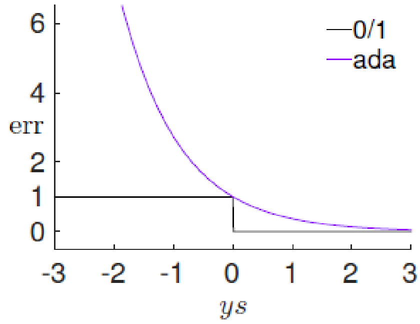
AdaBoost Error Function

claim: AdaBoost **decreases** $\sum_{n=1}^{N} u_n^{(t)}$ and thus somewhat **minimizes**

$$\sum_{n=1}^{N} u_n^{(T+1)} = \frac{1}{N}\sum_{n=1}^{N} \exp\left(-y_n \sum_{t=1}^{T} \alpha_t g_t(\mathbf{x}_n)\right)$$

linear score $s = \sum_{t=1}^{T} \alpha_t g_t(\mathbf{x}_n)$

- $\mathrm{err}_{0/1}(s, y) = [\![ys \le 0]\!]$
- $\widehat{\mathrm{err}}_{\mathrm{ADA}}(s, y) = \exp(-ys)$: upper bound of $\mathrm{err}_{0/1}$ —called **exponential error measure**



$\widehat{\mathrm{err}}_{\mathrm{ADA}}$: **algorithmic error measure** by **convex upper bound** of $\mathrm{err}_{0/1}$

$\widehat{err}_{ADA}(s,y) = e^{-ys}$ exponential error measure，它作为$err_{0/1}$的一个upper bound可以作为error function的度量

3. 对于$\widehat{err}_{ADA}(s,y) = e^{-ys}$我们也希望通过gradient descent的方法来减少这个误差。但是这个时候我们不知道对于权重的梯度到底是多少，我们先照搬一下：

Gradient Descent on AdaBoost Error Function

recall: gradient descent (**remember? :-)**), at iteration t

$$\min_{\|\mathbf{v}\|=1} E_{\mathrm{in}}(\mathbf{w}_t + \eta\mathbf{v}) \approx \underbrace{E_{\mathrm{in}}(\mathbf{w}_t)}_{\text{known}} + \underbrace{\eta}_{\text{given positive}} \mathbf{v}^T \underbrace{\nabla E_{\mathrm{in}}(\mathbf{w}_t)}_{\text{known}}$$

at iteration t, to find g_t, solve

$$\begin{aligned}
\min_h \quad \hat{E}_{\mathrm{ADA}} &= \frac{1}{N}\sum_{n=1}^{N} \exp\left(-y_n\left(\sum_{\tau=1}^{t-1} \alpha_\tau g_\tau(\mathbf{x}_n) + \eta h(\mathbf{x}_n)\right)\right) \\
&= \sum_{n=1}^{N} u_n^{(t)} \exp\left(-y_n \eta h(\mathbf{x}_n)\right) \\
&\overset{\text{taylor}}{\approx} \sum_{n=1}^{N} u_n^{(t)} \left(1 - y_n \eta h(\mathbf{x}_n)\right) = \sum_{n=1}^{N} u_n^{(t)} - \eta \sum_{n=1}^{N} u_n^{(t)} y_n h(\mathbf{x}_n)
\end{aligned}$$

good h: minimize $\sum_{n=1}^{N} u_n^{(t)} \left(-y_n h(\mathbf{x}_n)\right)$

我们加入一个$\eta h(x_n)$相当于走了一小步，就像梯度下降中向梯度方向走的那一步，我们做代数变换，以及泰勒展开得到了是

$$\widehat{E}_{ADA} \approx \sum_{n=1}^{N} u_n^{(t)} - \eta \sum_{n=1}^{N} u_n^{(t)} y_n h(x_n) \tag{1}$$

于是问题转换为找到 **good h (direction of function)**

$$s.t. minimize \sum_{n=1}^{N} u_n^{(t)} (-y_n h(x_n)) \tag{2}$$

函数的方向就类似于梯度方向。接着我们做代数变形

finding good h (function direction) $\Leftrightarrow$ minimize $\sum_{n=1}^{N} u_n^{(t)} (-y_n h(\mathbf{x}_n))$

for binary classification, where y_n and $h(\mathbf{x}_n)$ both $\in \{-1, +1\}$:

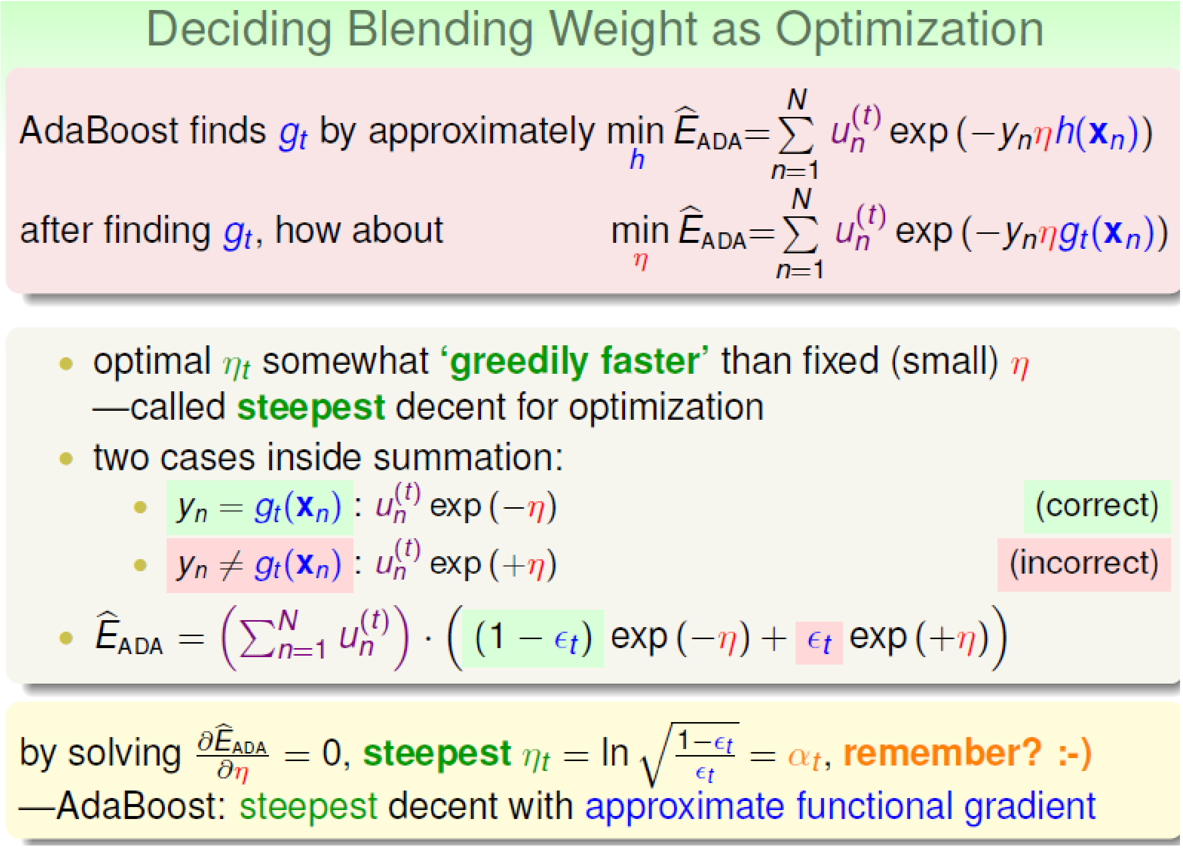
$$\begin{aligned}
\sum_{n=1}^{N} u_n^{(t)} (-y_n h(\mathbf{x}_n)) &= \sum_{n=1}^{N} u_n^{(t)} \begin{cases} -1 & \text{if } y_n = h(\mathbf{x}_n) \\ +1 & \text{if } y_n \neq h(\mathbf{x}_n) \end{cases} \\
&= -\sum_{n=1}^{N} u_n^{(t)} + \sum_{n=1}^{N} u_n^{(t)} \begin{cases} 0 & \text{if } y_n = h(\mathbf{x}_n) \\ 2 & \text{if } y_n \neq h(\mathbf{x}_n) \end{cases} \\
&= -\sum_{n=1}^{N} u_n^{(t)} + 2E_{\text{in}}^{\mathbf{u}^{(t)}}(h) \cdot N
\end{aligned}$$

—who minimizes $E_{\text{in}}^{\mathbf{u}^{(t)}}(h)$? $\mathcal{A}$ **in AdaBoost! :-)**

$\mathcal{A}$: **good** $g_t = h$ for 'gradient descent'

那么现在我们就需要让$E_{in}^{u^{(t)}}(h)$越来越小就行了，于是我们需要做的就是利用AdaBoost中的A降低它即可，我们通过演算法得到的$g_t = h$就是我们希望得到的结果，于是我们以此降低了$\widehat{E}_{ADA}$

4. 设置blending的权重：



我们首先通过最小化$\widehat{E}_{ADA} = \sum_{n=1}^{N} u_n^{(t)} e^{-y_n \eta h(x_n)}$来找到最好的$h = g_t$，也就是**function direction**紧接着我们希望利用这个学到的g_t来得到相应的η（事实上刚才我们并没有引入η这个参数，或者说$\eta = 1$），接下来就是需要找到一个η_t，它每一次都找到下降最快的位置。这个方法被称为是**steepest decent for optimization**也就是所谓的**"最速下降法"**

我们分析一下这个求和式，根据$y_n, g_t(x_n)$的关系可以分成两种情况，分别计算，其中:

$$y_n = g_t(x_n) : u_n^{(t)} e^{-\eta} \quad (correct) \tag{3}$$

$$y_n \neq g_t(x_n) : u_n^{(t)} e^{+\eta} \quad (incorrect) \tag{4}$$

$$\widehat{E}_{ADA} = (\sum_{n=1}^{N} u_n^{(t)}) * ((1-\epsilon_t)\, e^{-\eta} + \epsilon_t\, e^{+\eta}) \tag{5}$$

通过对η求导，我们得到了如下的式子，注意结果正好是我们之前使用的α_t:

$$\eta_t = ln\sqrt{\frac{1-\epsilon_t}{\epsilon_t}} = \alpha_t \tag{6}$$

因此AdaBoost可以看作是近似函数梯度的最速下降法！

三、Gradient Boosting

我们把AdaBoost的思想进行进一步的推广，我们希望得到更加普适的结论，在AdaBoost中我们的h(x)是二元分类的假设函数，输出是二值的，现在我们不对h(x)作任何限制(可以是回归可以是分类)

Gradient Boosting for Arbitrary Error Function

AdaBoost

$$\min_{\eta} \min_{h} \frac{1}{N}\sum_{n=1}^{N} \exp\left(-y_n\left(\sum_{\tau=1}^{t-1}\alpha_\tau g_\tau(\mathbf{x}_n)+\eta h(\mathbf{x}_n)\right)\right)$$

with binary-output hypothesis h

GradientBoost

$$\min_{\eta} \min_{h} \frac{1}{N}\sum_{n=1}^{N} \mathrm{err}\left(\sum_{\tau=1}^{t-1}\alpha_\tau g_\tau(\mathbf{x}_n)+\eta h(\mathbf{x}_n), y_n\right)$$

with any hypothesis h (usually real-output hypothesis)

GradientBoost: allows **extension to different** err for regression/soft classification/etc.

Regression: MSE

GradientBoost for Regression

$$\min_{\eta} \min_{h} \frac{1}{N}\sum_{n=1}^{N} \mathrm{err}\Big(\underbrace{\sum_{\tau=1}^{t-1}\alpha_\tau g_\tau(\mathbf{x}_n)}_{s_n}+\eta h(\mathbf{x}_n), y_n\Big)$$ with $\mathrm{err}(s,y)=(s-y)^2$

$$\min_{h} \ldots \overset{\text{taylor}}{\approx} \min_{h} \frac{1}{N}\sum_{n=1}^{N}\underbrace{\mathrm{err}(s_n,y_n)}_{\text{constant}}+\frac{1}{N}\sum_{n=1}^{N}\eta h(\mathbf{x}_n)\left.\frac{\partial \mathrm{err}(s,y_n)}{\partial s}\right|_{s=s_n}$$

$$= \min_{h} \ \text{constants}+\frac{\eta}{N}\sum_{n=1}^{N} h(\mathbf{x}_n)\cdot 2(s_n-y_n)$$

naïve solution $h(\mathbf{x}_n)=-\infty\cdot(s_n-y_n)$
if no constraint on h

我们首先让min_h项最小，通过泰勒展开配合对于voting score的求导我们得到了
$h(x_n)=-\infty*2(s_n-y_n)$

在对于h没有任何限制的情况下我们可以得到这个解。显然还不太恰当。我们看看在有条件限制情况下它的表现

事实上h的量级不是十分重要，因为我们之后学习的是η,接下来我们为了避免naive solution我们采取正则化的操作，就是不希望h太大：

Learning Hypothesis as Optimization

$$\min_{h} \quad \text{constants} + \frac{\eta}{N}\sum_{n=1}^{N} 2h(\mathbf{x}_n)(s_n - y_n)$$

- magnitude of h does not matter: because η will be optimized next
- penalize large magnitude to avoid naïve solution

$$\begin{aligned}\min_{h} \quad & \text{constants} + \frac{\eta}{N}\sum_{n=1}^{N}\left(2h(\mathbf{x}_n)(s_n - y_n) + (h(\mathbf{x}_n))^2\right) \\ = \quad & \text{constants} + \frac{\eta}{N}\sum_{n=1}^{N}\left(\text{constant} + (h(\mathbf{x}_n) - (y_n - s_n))^2\right)\end{aligned}$$

- solution of penalized approximate functional gradient: squared-error regression on $\{(\mathbf{x}_n, \underbrace{y_n - s_n}_{\text{residual}})\}$

GradientBoost for regression:
find $g_t = h$ by regression with **residuals**

通过化简，我们看出实际上我们就是求对于余数 residual的MSE误差。简言之，对于regresion的GradientBoost就是对于residuals来说MSE的最小的$g_t = h$

搞定了g_t我们考虑怎么学习η，这个时候实际上就是一个对于 residual 的linear regression:

Deciding Blending Weight as Optimization

after finding $g_t = h$,

$$\min_{\eta} \cancel{\min_{h}} \frac{1}{N}\sum_{n=1}^{N} \operatorname{err}\left(\underbrace{\sum_{\tau=1}^{t-1}\alpha_\tau g_\tau(\mathbf{x}_n)}_{s_n} + \eta g_t(\mathbf{x}_n), y_n\right) \text{ with } \operatorname{err}(s, y) = (s - y)^2$$

$$\min_{\eta} \quad \frac{1}{N}\sum_{n=1}^{N}(s_n + \eta g_t(\mathbf{x}_n) - y_n)^2 = \frac{1}{N}\sum_{n=1}^{N}((y_n - s_n) - \eta g_t(\mathbf{x}_n))^2$$

—one-variable linear regression on $\{(g_t\text{-transformed input}, \textbf{residual})\}$

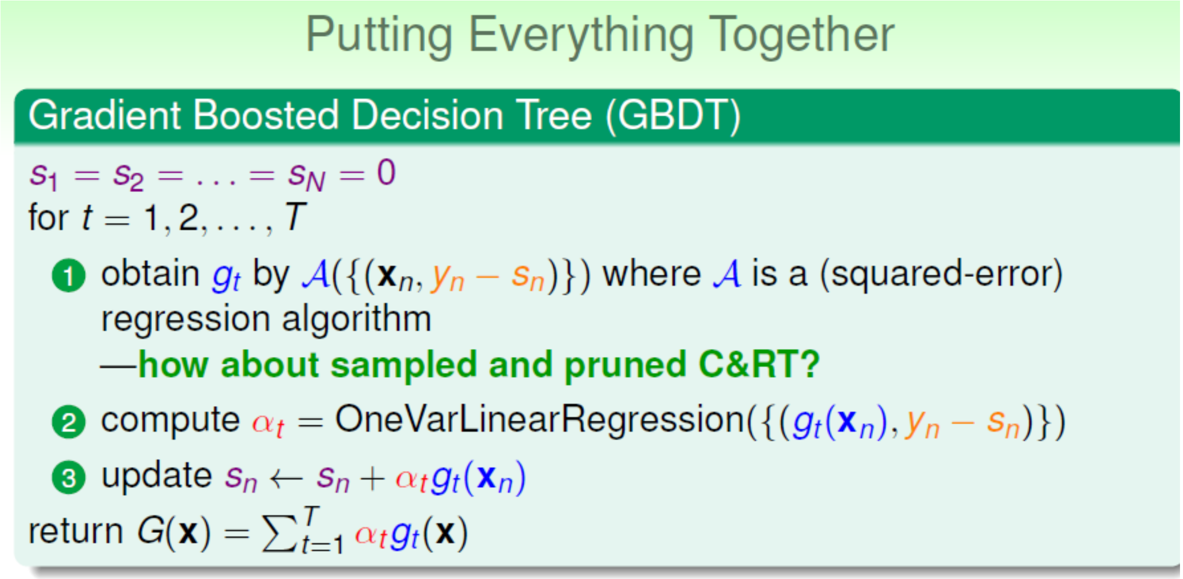
GradientBoost for regression: α_t = optimal η
by g_t-transformed linear regression

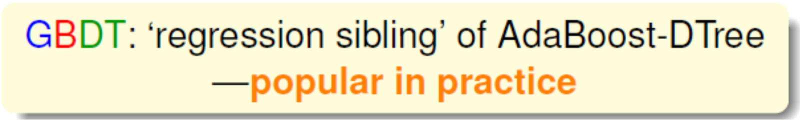
因此，GB for regression就是找到最好的 $\alpha_t = \eta$，方式是通过g_t转化的linear regression。

关于它的解：

$$\left(\sum_{n=1}^{N} g_t(\mathbf{x}_n)(y_n - s_n)\right) \Big/ \left(\sum_{n=1}^{N} g_t^2(\mathbf{x}_n)\right)$$

下面对于GDBT做一个总结：





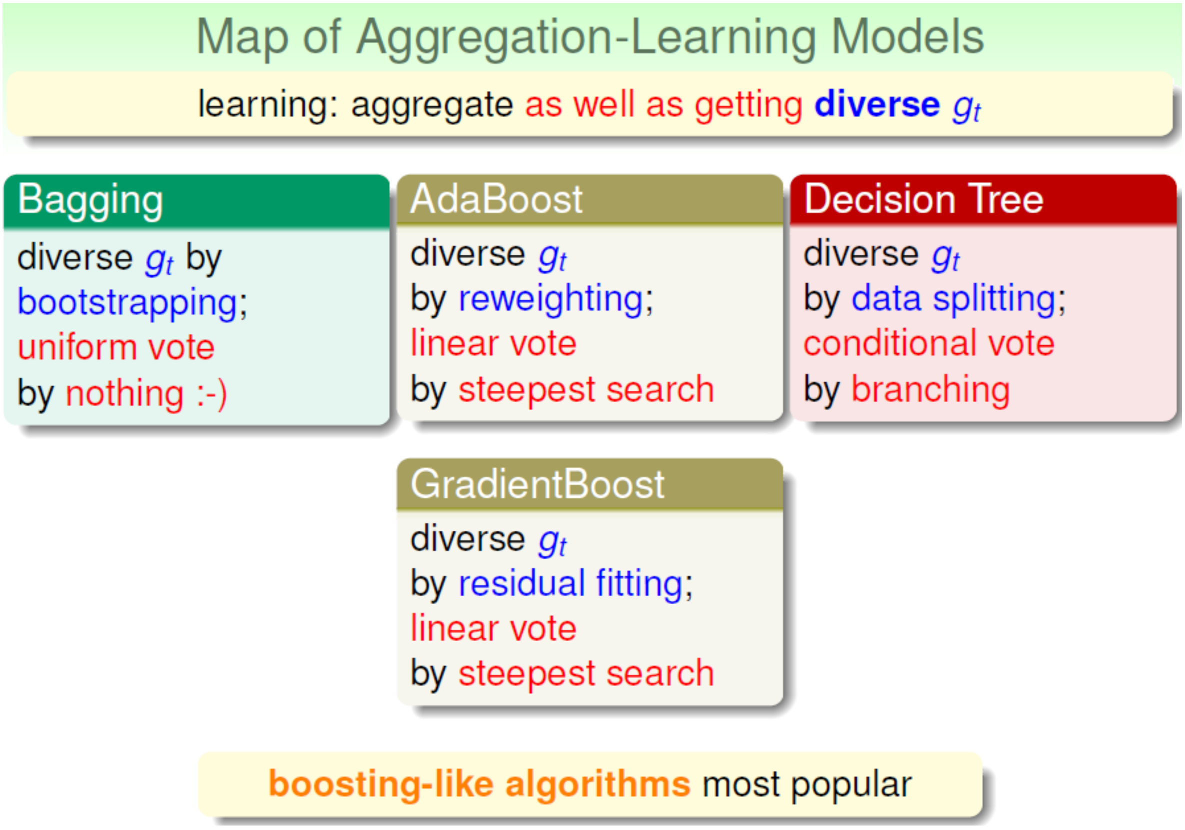
1. 通过回归算法获g_t使得$MSE(X_n, y_n - s_n)$最小
2. 利用单变量线性回归得到 $\alpha_t = LR(g_t(x_n), y_n - s_n)$
3. 更新voting score $s_n = s_n + \alpha_t g_t(x_n)$
4. 重复若干次
5. 返回 $G(x) = \sum_{t=1}^{T} \alpha_t g_t(x)$

GBDT在回归中用的特别多，和AdaBoost-DTree是“回归兄弟”

四、Summary of Aggregation Models

下面对于各种模型进行一个总结：

首先是Aggregation-learning中的几个基本类型，实际上boosting类型的算法是最为管饭应用的



其次是Aggregation of Aggregation的几个模型，在前人的基础上再创新高，三者应用都很广泛。