Boosting

对于一个复杂的决策树，它拥有低偏差和高方差。Bagging 的思路是构建多个模型来降低方差。而我们的新的思路是减少简单树的偏差使它们更具表现力 – boosting。

和 bagging 不同，bagging 是最小化方差，boosting 是最小化偏差。

Ensemble methods that minimize bias：

- Functional Gradient Descent
- Boosting
- Adboost

Boosting Algorithms

Boosting 背后的关键直觉是，人们可以采用简单模型 $\{T_h\}_{h\in H}$ 的集成，并将它们加法地组合成一个更复杂的模型。

$$T = \sum_h \lambda_h T_H$$

其中每个模型 T_h 可能不能很好的预测数据（高偏差），但集成的线性组合可以是富有表现力/灵活性的。

GRADIENT BOOSTING

Gradient boosting 是一种通过添加简单模型来迭代构建复杂回归模型 T 的方法。添加到集成中的每个新的简单模型都弥补了当前集成的弱点。

1. Fit a simple model $T^{(0)}$ on the training data

$$\{(x_1, y_1), \ldots, (x_N, y_N)\}$$

Set $T \leftarrow T^{(0)}$. Compute the residuals $\{r_1, \ldots, r_N\}$ for T.

2. Fit a simple model, $T^{(1)}$, to the current ***residuals***, i.e. train using

$$\{(x_1, r_1), \ldots, (x_N, r_N)\}$$

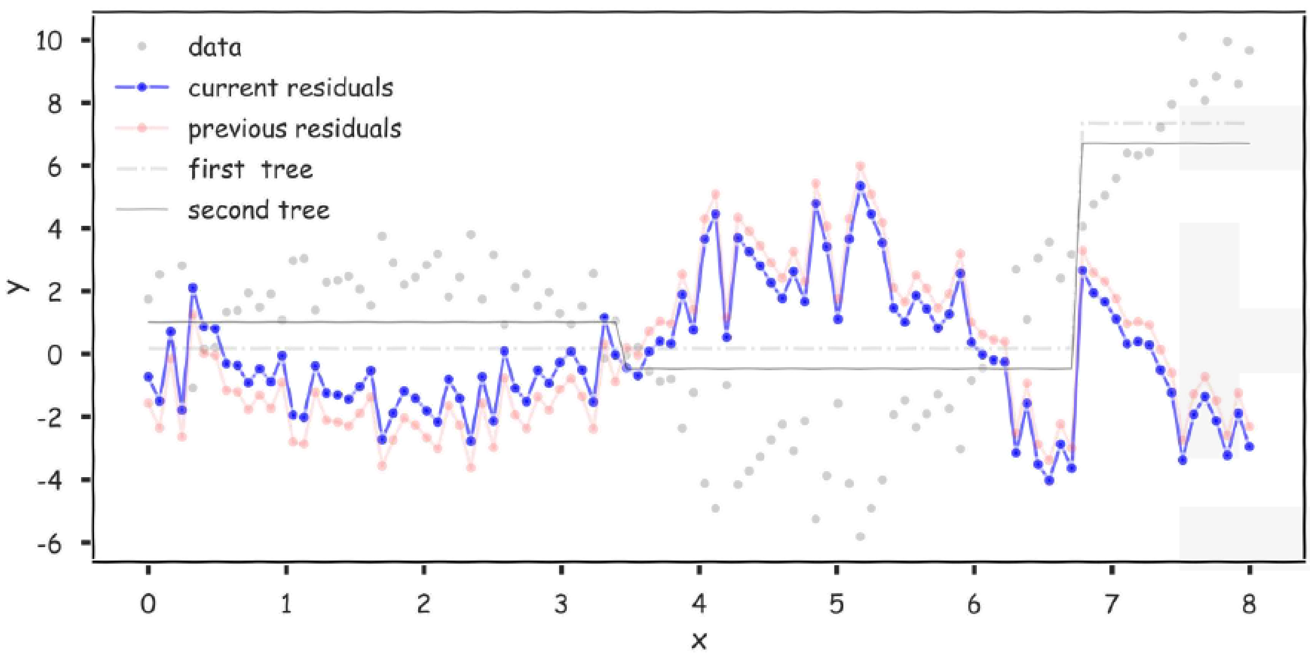
3. Set $T \leftarrow T + \lambda T^{(1)}$

4. Compute residuals, set $r_n \leftarrow r_n - \lambda T^{i}(x_n),\ n = 1, \ldots, N$

5. Repeat steps 2-4 until ***stopping*** condition met.

where λ is a constant called the ***learning rate***.

注：大概过程就是：先训一个模型，然后用该模型对训练集预测，再求出真实值和预测值之间的 residuals (r_n)；用 residuals 代替训练集中的真实值，再训练新的模型，最后将模型进行加权相加。



WHY DOES GRADIENT BOOSTING WORK?

每个简单模型 $T^{(i)}$ 都是对集成模型 T 的误差的建模。这样，我们把 $T^{(i)}$ 添加到集成模型 T 中，residual 会减少。

$$r_n - \lambda T^{(i)}(x_n)$$

现在我们的目的就是让 residual 最小化，这就是一个优化问题，可以使用 gradient descent 来进行。

GRADIENT BOOSTING AS GRADIENT DESCENT

- Often in regression, our objective is to minimize the MSE

$$\text{MSE}(\hat{y}_1, \ldots, \hat{y}_N) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

- Treating this as an optimization problem, we can try to directly minimize the MSE with respect to the predictions

$$\begin{aligned}\nabla \text{MSE} &= \left[\frac{\partial \text{MSE}}{\partial \hat{y}_1}, \ldots, \frac{\partial \text{MSE}}{\partial \hat{y}_N}\right] \\ &= -2\left[y_1 - \hat{y}_1, \ldots, y_N - \hat{y}_N\right] \\ &= -2\left[r_1, \ldots, r_N\right]\end{aligned}$$

- The update step for gradient descent would look like

$$\hat{y}_n \leftarrow \hat{y}_n + \lambda r_n, \quad n = 1, \ldots, N$$

注：λr_n 是我们得到的梯度，我们会根据梯度来训练新的模型。

CHOOSING A LEARNING RATE

在理想条件下，梯度下降以迭代方式近似并收敛到最优。

但是，我们应该在什么时候终止梯度下降：

- 我们可以限制下降过程中的迭代次数。但是，对于最大迭代的任意选择，我们不能保证我们最终足够接近最佳
- 如果在更新足够小时停止下降（例如，T的残差很小），我们会遇到一个新问题：算法可能永远不会终止

这两个问题都与学习速率 λ 的大小有关。

对于恒定学习速率 λ，如果 λ 太小，则需要多次迭代才能达到最优值。如果 λ 太大，算法可能会在最佳值附近"震荡"，并且永远不会足够接近。

如何选择 λ:

- 如果 λ 是一个常数，那么它应该通过交叉验证来调整
- 对于更好的结果，让 λ 成为一个变量，它的值依赖于梯度

$$\lambda = h(\|\nabla f(x)\|)$$

  - 其中，$\|\nabla f(x)\|$ 是梯度 $\nabla f(x)$ 的大小。因此在最优值附近时，梯度小，λ 也小；离最优值远时，梯度大，λ 也大。

AdaBoost

COMPONENTS: DECISION STUMPS

AdaBoost 的每棵树都只有一个根节点和两个叶子节点，因此也叫决策树桩 (decision stump)。

对于一个二分类问题 (+1/-1)，在 adaBoost 中会生成一系列的分量预测器 (component classifiers):

$$h(\mathbf{x};\theta) = sign(w_1 x_k - w_0)$$

其中 $\theta = \{k, w_1, w_0\}$。因为 adaBoost 的每棵树都只有一个根节点和两个叶子节点，因此每个树只能对一个输入向量的一个分量进行预测 (比如下面的胸口疼痛)，x_k 就是其中的一个分量。

GRADIENT DESCENT WITH EXPONENTIAL LOSS

我们使用 exponential loss 来计算 adaBoost 的分类误差:

$$\text{ExpLoss} = \frac{1}{N}\sum_{n=1}^{N} \exp\left(-y_n \hat{y}_n\right), y_n \in \{-1, 1\}$$

- We first compute the gradient for ExpLoss:

$$\nabla \text{Exp} = [-y_1 \exp(-y_1 \hat{y}_1), \ldots, -y_N \exp(-y_N \hat{y}_N)]$$

- It's easier to decompose each $y_n \exp(-y_n \hat{y}_n)$ as $w_n y_n$, where

$$w_n = \exp(-y_n \hat{y}_n) .$$

- This way, we see that the gradient is just a re-weighting applied the target values

$$\nabla \text{Exp} = [-w_1 y_1, \ldots, -w_N y_N]$$

- Notice that when $y_n = \hat{y}_n$, the weight w_n is small; when $y_n \neq \hat{y}_n$, the weight is larger.
- The update step in the gradient descent is

$$\hat{y}_n \leftarrow \hat{y}_n + \lambda w_n y_n, \qquad n = 1, \ldots, N$$

和 gradient boosting 一样，我们训练新的决策树桩近似梯度 $\lambda w_n y_n$。

- This means training $T^{(i)}$ on a re-weighted set of target values,

$$\{(x_1, w_1 y_1), \ldots, (x_N, w_N y_N)\}$$

也就是说，我们使用 exponential loss 的梯度下降来迭代训练新的简单模型，而该模型侧重于先前模型错误分类的点。

AdaBoost 中会有很多弱模型，接下来，我们需要定义一个损失函数，以便我们可以确定要添加的新分量 h(x; θ) (adaBoost 会一个一个训练新的分量模型，并将其加入总模型)，以及它占的权重。

$$h_m(\mathbf{x}) = \alpha_1 h(\mathbf{x}; \theta_1) + \ldots + \alpha_m h(\mathbf{x}; \theta_m)$$

虽然损失函数有很多选项，但我们在这里只考虑一个简单的指数损失:

$$exp\{-y\, h_m(\mathbf{x})\}$$

- Consider adding the m^{th} component:

$$\begin{aligned}
&\sum_{i=1}^{n} \exp\{\, -y_i[h_{m-1}(\mathbf{x}_i) + \alpha_m h(\mathbf{x}_i;\theta_m)]\,\} \\
&\quad = \sum_{i=1}^{n} \exp\{\, -y_i h_{m-1}(\mathbf{x}_i) - y_i \alpha_m h(\mathbf{x}_i;\theta_m)\,\} \\
&\quad = \sum_{i=1}^{n} \underbrace{\exp\{\, -y_i h_{m-1}(\mathbf{x}_i)\}}_{\text{fixed at stage } m} \exp\{-y_i \alpha_m h(\mathbf{x}_i;\theta_m)\,\} \\
&\quad = \sum_{i=1}^{n} W_i^{(m-1)} \exp\{-y_i \alpha_m h(\mathbf{x}_i;\theta_m)\,\}
\end{aligned}$$

注：上式就是 $exp\{-y\ h_m(\mathbf{x})\}$ 的展开。

现在我们就要解决两个问题，一个是要添加的新分量 h(x; θ)，另一个是它占的权重 α。

我们先将 α 从式子中消除，以方便找到合适的 h(x; θ)。

$$\begin{aligned}
&\frac{\partial}{\partial \alpha_m}\Big|_{\alpha_m=0} \sum_{i=1}^{n} W_i^{(m-1)} \exp\{-y_i \alpha_m h(\mathbf{x}_i;\theta_m)\,\} = \\
&\quad \left[\sum_{i=1}^{n} W_i^{(m-1)} \exp\{-y_i \alpha_m h(\mathbf{x}_i;\theta_m)\,\} \cdot \big(-y_i h(\mathbf{x}_i;\theta_m)\big)\right]_{\alpha_m=0} \\
&\quad = \left[\sum_{i=1}^{n} W_i^{(m-1)} \big(-y_i h(\mathbf{x}_i;\theta_m)\big)\right]
\end{aligned}$$

之后找到能最小化 error 的 h(x; θ)。

- We find $h(\mathbf{x}; \hat{\theta}_m)$ that minimizes

$$-\sum_{i=1}^{n} \tilde{W}_i^{(m-1)} y_i h(\mathbf{x}_i; \theta_m)$$

where $\sum_{i=1}^{n} \tilde{W}_i^{(m-1)} = 1$.

再得到分量模型所占的权重 α。

- α_m is subsequently chosen to minimize

$$\sum_{i=1}^{n} \tilde{W}_i^{(m-1)} \exp\{-y_i \alpha_m h(\mathbf{x}_i; \hat{\theta}_m)\}$$

总的过程如下：

Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in \mathscr{X}$, $y_i \in \{-1, +1\}$.

Initialize $\boxed{D_1(i) = 1/m \text{ for } i = 1, \ldots, m.}$ Initial Distribution of Data

For $t = 1, \ldots, T$:

- Train weak learner using distribution D_t.
- Get weak hypothesis $h_t : \mathscr{X} \to \{-1, +1\}$. Train model
- Aim: select h_t with low weighted error:

$$\boxed{\varepsilon_t = \Pr_{i \sim D_t}[h_t(x_i) \neq y_i].}$$ Error of model

- Choose $\boxed{\alpha_t = \frac{1}{2} \ln\left(\frac{1 - \varepsilon_t}{\varepsilon_t}\right)}$. Coefficient of model
- Update, for $i = 1, \ldots, m$:

$$\boxed{D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}}$$ Update Distribution

where Z_t is a normalization factor (chosen so that D_{t+1} will be a distribution).

Output the final hypothesis:

$$\boxed{H(x) = \operatorname{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right).}$$ Final average

训练误差呈指数级快速下降。

CHOOSING THE LEARNING RATE

与回归的梯度提升不同，我们可以通过优化来解析解决 adaBoost 的最佳学习率：

$$\underset{\lambda}{\text{argmin}} \frac{1}{N} \sum_{n=1}^{N} \exp\left[-y_n(T + \lambda^{(i)} T^{(i)}(x_n))\right]$$

- Doing so, we get that

$$\lambda^{(i)} = \frac{1}{2} \ln \frac{1-\epsilon}{\epsilon}, \quad \epsilon = \sum_{n=1}^{N} w_n \mathbb{1}(y_n \neq T^{(i)}(x_n))$$

注：T 是 residuals。