Extreme Gradient Boosting for Regression and Classification

--Summary and Understanding from StatQuest

Created by *Ronghzeng (Roger) He*

	Regression	**Classification**	
	Solution, Action	Solution, Action	Understanding
Input data $\{(x_i, y_i)\}_{i=1}^{n}$, and a differentiable **Loss Function** $L(y_i, F(x_i))$ Similarity function (score) for Decision Tree, with λ the parameter for regularization $\mathrm{SS}(y_i, F(x_i) + \gamma) =$ $-2 \times \left[L(y_i, F(x_i) + \gamma) + \frac{1}{2}\lambda\,\gamma^2\right]$ Pruning function f_{prune} $2\nu T - \mathrm{SS}(y_i, F(x_i) + \gamma)$	$F(x)$ is a weak model, **Loss function** uses scaled RRS $\mathrm{L} = \frac{1}{2}\sum_i [y_i - F(x_i)]^2$	**Loss function** uses cross entropy $-\sum_i [p_i ln(\hat{p}_i) + (1 - p_i) ln(1 - \hat{p}_i)]$ **log(odds)**: $ln(odds) = -ln\left(\frac{1-p}{p}\right)$ **Probability**: $p = \frac{1}{1+e^{-log(odds)}}$	Using those forms of functions are for later convenience; For classification, $F(x_i)$ is log(odds), log(odds) calculated by different models can add up together, but probabilities cannot.
Step1: the 0th model is set to be 0.5, by default $F_0(x) = 0.5$	By default, it's 0.5, but of course you can use a value that minimizes the loss function		
Step2: for m=1 to M Where m denotes mth model	Iterate through weak models one by one, until a max number of models is reached or additional models don't improve the fit.		
Step2.1: compute residuals $\gamma_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x_i)=F_{m-1}(x_i)}$	$\gamma_{im} = y_i - F_{m-1}(x_i)$	$\gamma_{im} \approx \mathrm{p_i} - \hat{\mathrm{p}}_{m-1,i}$, when γ is small. $\hat{\mathrm{p}}_{m-1,i}$ is the predicted probability that ith data point is positive, by model $F_{m-1}(x_i)$.	residuals for each data point, indicating where the previous model creates errors.

Step2.2: Fit a **decision tree** to γ_{im} values, splitting on features based on **similarity score** (ss) with optimal output value γ at each decision node. Denotes each leaf by R_{jm}, for all leaves in m[th] tree.	Similarity score for each decision node $= \frac{[\sum(\gamma_{im} \in R_{jm})]^2}{\sum \delta(\mathrm{x_i} \in \mathrm{R_{jm}}) + \lambda}$ Min number of values in a node to go on splitting, $\mathrm{N_{min}}$, is 1	Similarity score for each decision node $= \frac{[\sum(\gamma_{im} \in R_{jm})]^2}{\sum_{\mathrm{x_i} \in \mathrm{R_{jm}}}[\hat{\mathrm{p}}_{m-1,i} \times (1 - \hat{\mathrm{p}}_{m-1,i})] + \lambda}$ $\mathrm{N_{min}} = \sum_{\mathrm{x_i} \in \mathrm{R_{jm}}}[\hat{\mathrm{p}}_{m-1,i} \times (1 - \hat{\mathrm{p}}_{m-1,i})]$	For each split, always find the feature that increases the similarity, or equivalently decreases the (Loss_function + regularization term)
Pruning: after building a full tree, we can start pruning to keep the balance between tree complexity and (loss + regularization)	• T denotes a single tree model complexity, e.g. the total number of leaves in a tree. • For each split of a decision node, the similarity increased by ΔSS and model complexity increased by $2\nu\Delta T$, but we want SS to be large and model complexity small, or equivalently f_{prune} to be small, so there is a tradeoff b/w those two. • If $\Delta SS \geq 2\nu\Delta T$ at a node, then Δf_{prune} will be non-increasing, which is favorable.		
Step 2.3: For j[th] leaf in m[th] tree, compute $\gamma_{jm} =$ $arg\min_{\gamma} \sum_{\mathrm{x_i} \in \mathrm{R_{jm}}} L(y_i, F_{m-1}(x_i) + \gamma)$	Average of pseudo residuals that end up in the same leaf with regularization $\gamma_{jm} = \frac{\sum(\gamma_{im} \in R_{jm})}{\sum \delta(\mathrm{x_i} \in \mathrm{R_{jm}}) + \lambda}$	γ_{jm} $= \frac{\sum(\gamma_{im} \in R_{jm})}{\sum_{\mathrm{x_i} \in \mathrm{R_{jm}}}[\hat{\mathrm{p}}_{m-1,i} \times (1 - \hat{\mathrm{p}}_{m-1,i})] + \lambda}$ $\hat{\mathrm{p}}_{m-1,i}$ is the predicted prob by model $F_{m-1}(x_i)$.	To approximate the values in the same leaf by a constant, which is a solution to (Loss + regularization); iterate over all leaves.
Step 2.4: Update model $F_m(x) = F_{m-1}(x) + \eta \cdot \gamma_{jm} \cdot I(x \in R_{jm})$, Where η is a scaling factor for each model, similar to step size	Add up all scaled results from a series of weak models.	Add up all scaled log(odds) from a series of weak models. Still need to convert to probability.	This is why it's called an ensemble method.
Step 3: Output $F_M(x)$	For new input x, use model $F_M(x)$ to predict.	Convert the result from the last step to probability by $p = \frac{1}{1 + e^{-log(odds)}}$	May need to tune η to find the best converged result.