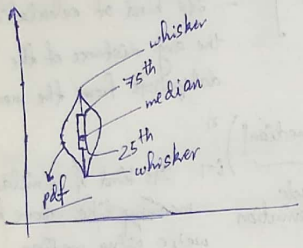
violin-plots → (box plot + pdf)



* Always perform analysis oriented towards solving the objective. Don't deviate.

Pseudo Residuals Revisited

* We have seen that pseudo-residuals $\left(-\frac{\partial L}{\partial F_k(x)}\right)$ are similar to → residuals/errors for squared loss function.

* What about other loss functions?

→ consider binary log-loss.

$$L = y_i \cdot \log(p_i) + (1-y_i) \cdot \log(1-p_i)$$ where

$$p_i = \frac{1}{1+e^{-\hat{y}_i}}$$

So, $$L = y_i \cdot \log\left(\frac{1}{1+e^{-\hat{y}_i}}\right) + (1-y_i)\log\left(\frac{e^{-\hat{y}_i}}{1+e^{-\hat{y}_i}}\right)$$

$$\frac{\partial L}{\partial \hat{y}} = \frac{y_i - (1-y_i)\cdot e^{\hat{y}_i}}{\left(1+e^{\hat{y}_i}\right)}$$ Multiplying by $\frac{e^{-\hat{y}_i}}{e^{-\hat{y}_i}}$

$$\frac{\partial L}{\partial \hat{y}} = \frac{y_i \cdot e^{-\hat{y}_i} + y_i - 1}{\left(1+e^{-\hat{y}_i}\right)} = \frac{y_i\left(e^{-\hat{y}_i}+1\right)}{\left(e^{-\hat{y}_i}+1\right)} - \frac{1}{\left(e^{-\hat{y}_i}+1\right)}$$

$$= y_i - p_i$$

$\hat{y}_i$ is same as $F_k(x)$

and $(y_i - p_i)$ is nothing but residue.

So, $\frac{-\partial L}{\partial F_k(x)} \propto$ residual

Hence, even for binary log-loss, the pseudo residuals are similar to the actual residuals.

sq-loss → yes. } most common loss.
log-loss → yes. }
any-loss → no

* So, if pseudo residuals aren't equivalent to residuals always, why are we even using them?

* It has a more fundamental mathematical reason behind this rather than pseudo residuals just being similar to residuals.

* We have — dataset $D = \{x_i, y_i\}_{i=1}^{n}$,

Loss function = L,

* we have to find a function F that maps $x_i \to y_i$ minimizing the loss function.

* In boosting we find a sequence of functions.

* So, in accordance to "emperical risk minimization" principle, we try to find $F(x)$ that minimizes average value of the loss function on the training set.

It does so by starting with a ~~constant~~ model, consisting of a constant function $F_0(x)$ and incrementally expanding in greedy fashion:-

$$F_0(x) = \underset{\gamma}{\operatorname{argmin}} \sum_{i=1}^{n} L(y_i, \gamma)$$

optimization Problem → $$F_m(x) = F_{m-1}(x) + \underset{h_m \in H}{\operatorname{argmin}} \sum_{i=1}^{n} L(y_i, F_{m-1}(x_i) + h_m(x_i))$$

H = set of all base learners possible.

Due to the infinite size of H, training the model isn't feasible.

So we try to find some approximations that are close to this optimization problem.

We solved the optimization problem of weights using gradient descent.

$$w_{new} = w_{old} - \eta \left(\frac{\partial L}{\partial w}\right)_{old}$$

update step.

variable — learning rate — gradient

Similarly we can generalize this for learning functions as well.

$$F_m(x) = F_{m-1}(x) - \gamma_m \sum_{i=1}^{n} \nabla_{F_{m-1}} L(y_i, F_{m-1}(x_i))$$

γ_m → Learning rate → calculated by simple line search.

this is nothing but pseudo-residual (including (-)ve sign).

Refer to prev notebook for actual algo of boosting using pseudo-residuals.

* So, this is the reason behind using pseudo-residuals.