Boosting (Chapter 18 of Prob ML book)

- Model's form is weighted sum of component models, like in bagging, but the training method is different $f(x) = \beta_1 F_1(x, \theta_1) + \dots \beta_M F_M(x, \theta_M)$
- Form of model:
  - $f(x;\theta) = \sum_1^M \beta_m F_m(x, \theta_m)$

$\beta_1 \quad \beta_2 \quad \beta_M$

$F_1(\theta_1) \quad F_2(\theta_2) \quad - - \quad F_M(\theta_m)$

$\uparrow x \quad \uparrow x \quad \uparrow x$

- Sequentially train weak learners where subsequent learners focus on residues of the previous stages → complicated decision boundaries from simple classifiers
- Simple classifier examples:
  - Naïve Bayes classifier
  - Decision stumps
  - Linear classifiers.

Forward stagewise additive boosting

- Sequentially optimize the overall training loss while keeping previous stages fixed.
- Given training data and loss: $f(x, \theta)$ $D = \{(x^i, y_i) : i = 1..N\}$

$$\min_{\theta} \sum_{i=1}^{N} l(y_i, \underbrace{f(x^i, \theta)}) = \min_{\theta_1, \theta_2, \theta_M} \sum_{i=1}^{N} l(y_i, \sum_{m=1}^{M} \beta_m F_m(x^i, \theta_m))$$

- Initially: $\boxed{f_1} = \operatorname{argmin}_\theta \sum_{i=1} l(y_i, f(x^i, \theta)) = \operatorname{argmin}_{\theta} \sum_{i=1}^{N} l(y_i, F_1(x^i, \theta))$
- For m = 2 to M
-

$$\theta_m = \operatorname{argmin}_{\theta} \sum_{i=1}^{N} l(y_i, \underbrace{f_{m-1}}(x^i) + F_m(x^i, \theta))$$

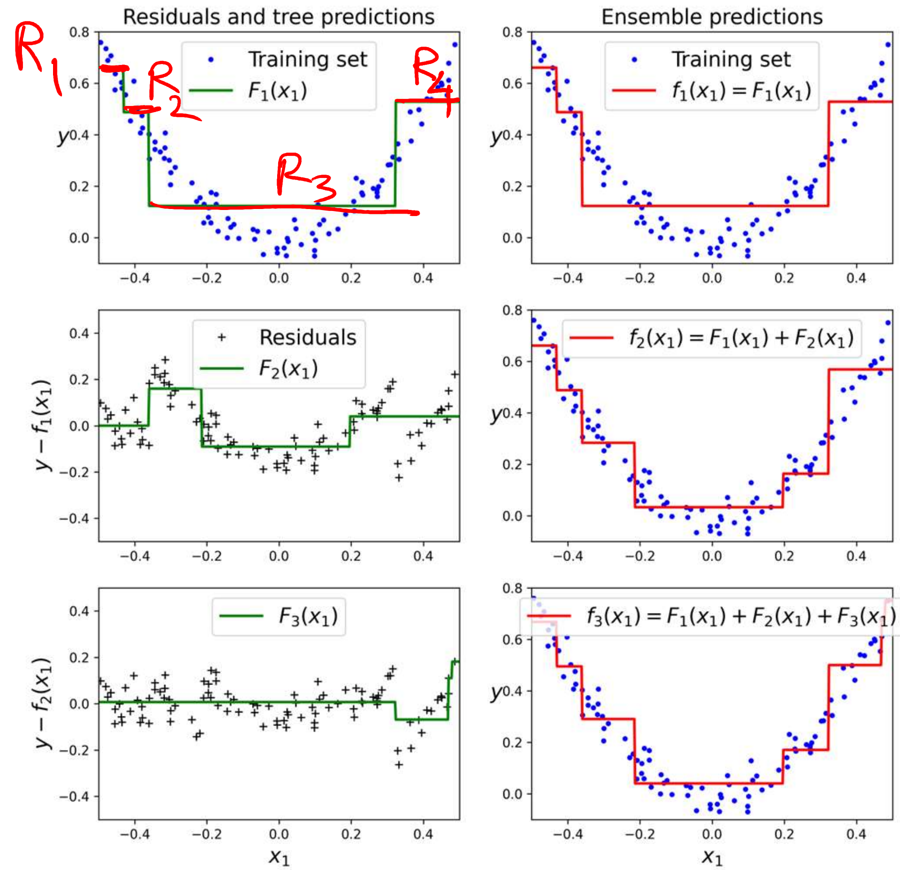
$$f_m = \sum_{j=1}^{m} F_j(x; \theta_j)$$

$f = f_M$

$\beta_m \Leftarrow$ Algorithm specific to be covered.
default $= 1$.

Boosted regression trees with square loss

- When loss function is square loss



- https://drive.google.com/file/d/1a7kfsrBtjwhP4x7XbwOtbWPG oxdP5WS/view?usp=sharing

Gradient boosting algorithm.

- A general-purpose algorithm for any differentiable loss function.
- Given N training examples, view $f(x^i)$ as an independent parameter

$w_1 = f(x^1)$ $w_2 = f(x^2)$ ... $w_N = f(x^N)$

$$\min_{f(x_1)\cdots f(x^N)} \sum_{i=1}^{N} \ell(y_i - f(x^i))^2$$

- At each stage m, we have a current value of the function.

$w_i^m = f_{m-1}(x^i)$ $f_{m-1}(x^i) = \sum_{j=1}^{m-1} \beta_j F_j(x^i)$

- Find gradient of the loss with respect to each $f(x^i)$ independently.
- $\cancel{g_{im}}$ γ_{im} $= \dfrac{\partial \sum_{i=1}^{N} \ell(y_i - f(x^i))}{\partial f(x^i)} \Bigg|_{f(x^i) = w_i^m = f_{m-1}(x^i)}$

- Function at the mth stage F_m is trained to match the gradients.

Algorithm 18.3: Gradient boosting

Initialize $f_0(\boldsymbol{x}) = \operatorname{argmin}_F \sum_{i=1}^N L(y_i, F(\boldsymbol{x}_i))$

for $m = 1 : M$ **do**

  Compute the gradient residual using $r_{im} = -\left[\frac{\partial L(y_i, f(\boldsymbol{x}_i))}{\partial f(\boldsymbol{x}_i)}\right]_{f(\boldsymbol{x}_i) = f_{m-1}(\boldsymbol{x}_i)}$

  Use the weak learner to compute $F_m = \operatorname{argmin}_F \sum_{i=1}^N (r_{im} - F(\boldsymbol{x}_i))^2$

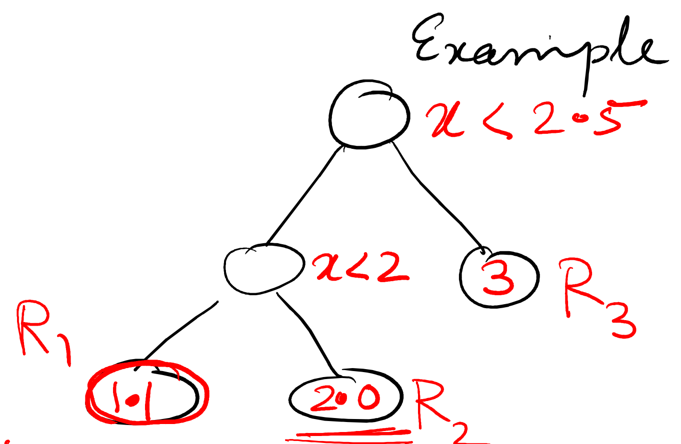
  Update $f_m(\boldsymbol{x}) = f_{m-1}(\boldsymbol{x}) + \nu F_m(\boldsymbol{x})$

Return $f(\boldsymbol{x}) = f_M(\boldsymbol{x})$

Gradient boosted regression trees.

- Assume tree is of the form

$$F_m(x) = \sum_{j=1}^{J_m} w_{jm} \mathbb{I}(x \in R_{jm})$$



- First find gradient of loss at m (residuals) and build the tree to find good regions

loss function = square loss e

$$\frac{\partial}{\partial f(x_i)} \frac{1}{2}\left[(y_i - f(x^i))^2\right] \Big| f(x^i) = f_{m-1}(x^i)$$

$$-(y_i - f(x^i)) \Big| f(x^i) = f_{m-1}(x^i)$$

→ residue

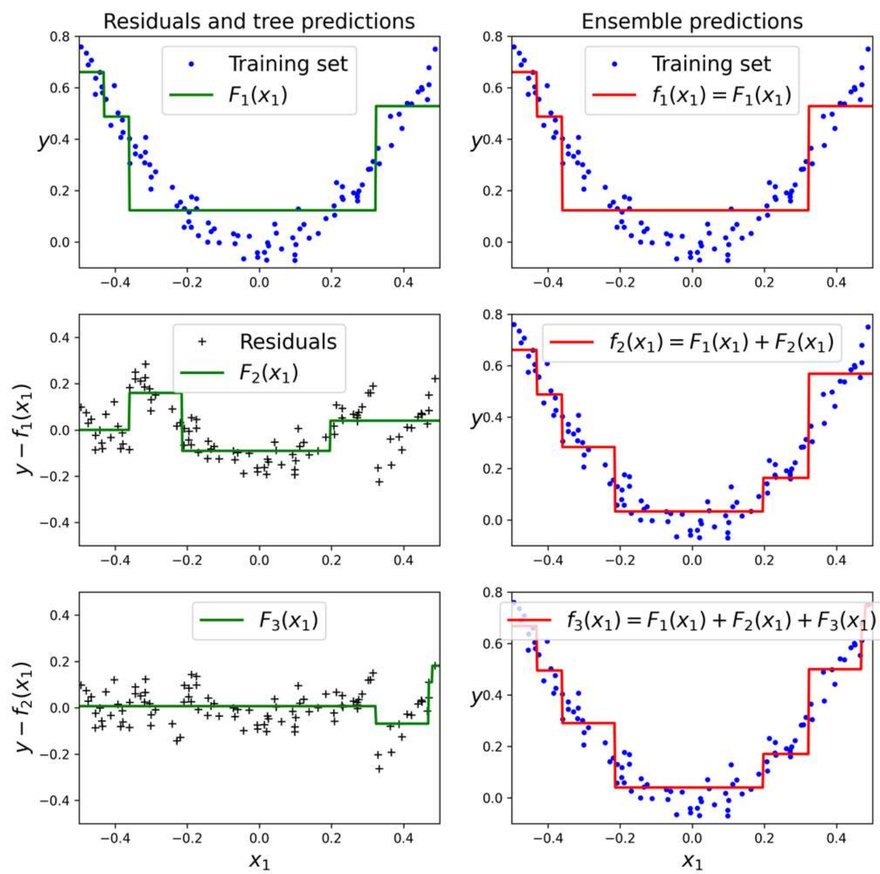
$$1.1 \cdot \mathbb{I}(x<2) + 2.0\, \mathbb{I}(2 \le x < 2.5) + 3\, \mathbb{I}(x \geq 2.5)$$

- Then resolve for the weights of each leaf by

$$\hat{w}_{jm} = \arg\min_{w} \sum_{x_i \in R_{jm}} \ell(y_i, f_{m-1}(x_i) + w)$$

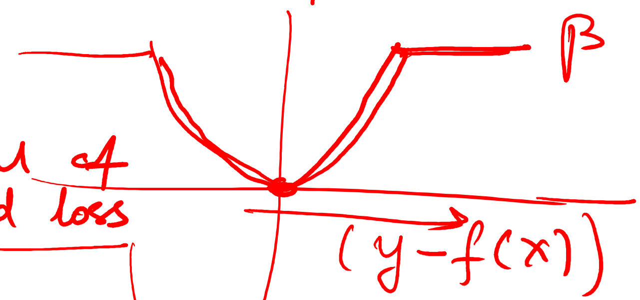
Boosted regression trees with square loss

- When loss function is square loss
    - Residue is:
    - Weight of each leaf
- [https://drive.google.com/file/d1a7kfsrBtjwhP4x7XbwOtbWPG oxdP5WS/view?usp=sharing](https://drive.google.com/file/d1a7kfsrBtjwhP4x7XbwOtbWPG)

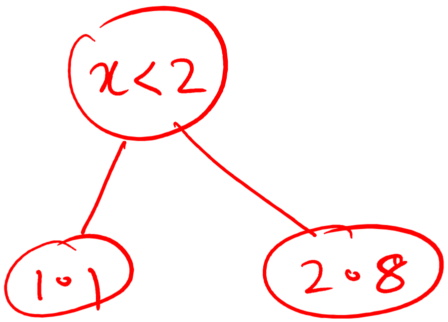


Fit regression tree with clipped loss

$$\min\left(\beta, \; (y-f(x))^2\right)$$



F_1



x	y	r = $y-F_1(x)$	gradient of clipped loss
1	1	-0.1	0.1
1.1	1.2	0.1	-0.1
2	2.1	-0.7	0
2.2	1.9	-0.9	0
4	3	0.2	-0.2
4.2	2.8	0	0

$\beta = 0.1$

$$\frac{\partial}{\partial f(x^i)} \min\left(\beta, \frac{(y-f(x^i))^2}{2}\right)$$

$$= 0 \text{ if } \frac{(y-f(x^i))^2}{2} \geq \beta$$

$$= (y-f(x^i))$$

Other variants

- Extreme gradient boosting (XGBoost).
- Significantly more evolved.
- But high accuracy with high speed training
- Extensively used