Bagging (Bootstrap aggregation)

Given: TS $S = \{(\underline{x}_1, y_1) \dots (x_n\, y_n)\}$ iid $\sim (\underline{X}, Y)$

Goal: Estimate $f(\underline{x}) = E(Y \mid \underline{x})$

$\hat{f}(\underline{x}, S)$ estimate for $f(\underline{x})$ based on training sample S (for example CART tree)

$S_1^* \dots S_B^*$ Bootstrap samples drawn from S (Bootstrap sample = random sample of size n drawn with replacement)

Bagged estimate

$$\hat{f}_{Bag}(\underline{x}, S) = \frac{1}{B} \sum_i \hat{f}(\underline{x}, S_i^*)$$

(Bogus) motivation

If we had B training samples of size n all drawn from $(\underline{X}, Y)$ then we could generate B prediction rules and average them

$S_1 \dots S_B$

$$\bar{f}(\underline{x}) = \frac{1}{B} \hat{f}(\underline{x}, S_i)$$

- $\bar{f}$ would have lower variance than $\hat{f}(\underline{x}, S)$
- Lacking additional training samples we use Bootstrap samples.

Empirical observation

Bagging can lead to improved performance for prediction rules that depend on the training sample in non-smooth way. (for example CART)

Conjecture: Bagging CART trees helps because bagged trees better approximate smooth responses.

Note: Interpretabity is lost

Note: Bagging is often applied to fully grown trees — no pruning

Note: Can use "out-of-bag" error estimation instead of CV

$$\hat{f}_{OOB}(\underline{x}_i, S) = \underset{j}{\text{ave}}\left(\hat{f}(\underline{x}_i, S_j^*) \,/\, (x_i, y_i) \notin S_j^*\right)$$

$$EPE = \frac{1}{n} \sum (y_i - \hat{f}_{OOB}(\underline{x}_i, S))^2$$

Boosting regression trees

Given: TS $(\underline{x}_1, y_1) \ldots (\underline{x}_n, y_n) \sim iid\ (\underline{X}, Y)$

Goal: Estimate $f(\underline{x}) = E(Y \mid \underline{x})$

Initially: $r_i = y_i \quad i = 1 \ldots n$

Define $T(\underset{\sim}{x}, \underset{\sim}{r})$ CART tree trained on $(\underset{\sim}{x}_1, r_1) \dots (\underset{\sim}{x}_n, r_n)$ with K splits (K usually $1..3$)

Boosting algorithm

For $i = 1 \dots M$ {

$$T_m(\underset{\sim}{x}) = T(\underset{\sim}{x}, \underset{\sim}{r}) \quad (1)$$

$$\underset{\sim}{r} = \underset{\sim}{r} - \alpha\left(T_m(\underset{\sim}{x}_1) \dots T_m(\underset{\sim}{x}_n)\right) \quad (2)$$

}

$$\hat{f}(\underset{\sim}{x}) = \alpha \sum_{m=1}^{M} T_m(\underset{\sim}{x})$$

α between 0.1 and 0.01

(1) Apply "base learner" to current residuals

(2) Update residuals

→ Build tree T_1 on $(\underset{\sim}{x}_1, y_1) \dots (\underset{\sim}{x}_n, y_n)$

→ Compute predicted values $T_1(\underset{\sim}{x}_1) \dots T_1(\underset{\sim}{x}_n)$

→ Form new "training data"

$(\underset{\sim}{x}_1, y_1 - \alpha T_1(x_1)) \dots (\underset{\sim}{x}_n, y_n - \alpha T_1(x_n))$.

Notes

$K = 1$ Base learner generates a tree with one split and two leaves ("stump")

$\Rightarrow \hat{f}(\underset{\sim}{x})$ will be additive with piece wise constant coordinate functions

⇒ Alternative way of learning additive model with piece wise constant coordinate functions:

- Define dictionary $\mathcal{B} = \mathcal{B}_1 \cup \mathcal{B}_2 \cup \mathcal{B}_p$
  $\mathcal{B}_j$ set of basis functions that depend only on predictor j

  $$\mathcal{B}_j = \{ B_{jk}(z) \quad k=1..n \}$$

  $$B_{jk}(z) = I(z_j \geq x_{kj})$$

Note: These functions are not linearly independent but that does not matter

We have such models before (for piece wise linear basis functions (using forward step wise regression.

What's different

- Boosting uses "stagewise", not "stepwise" regression.
  When a new basis function is added to the model, coefficients of previous basis functions are not readjusted.

- The least squares coef of the new basis function is multiplied by the learning rate.