9.6 Boosting

In bagging we fit many trees in parallel on bootstrapped datasets. Each tree might be quite deep and complicated. In boosting, we fit a small model and then another model that tries to fix the problems with the first. This procedure is iterated.

AdaBoost

AdaBoost was an extremely popular boosting method for classification (with many generalizations). We'll look at AdaBoost.M1. Suppose $Y \in \{-1, +1\}$ & Features are X.

Given a starting classifier $G_1(X)$ (outputs +1/-1), we develop a new $G_2(X)$ that concentrates on the points where $Y_i \neq G_1(X_i)$ (i.e. misclassification)

Algorithm for AdaBoost.M1

[n = # of data points]

- Initialize weights $w_i = \frac{1}{n}$ $i = 1, \ldots, n$
- For $m = 1, \ldots, M$
  - Fit classifier $\hat{G}_m(x)$ to training data using weights $\{w_i\}$
  - Compute

$$err_m = \frac{\sum_{i=1}^{n} w_i \, \mathbb{1}[y_i \neq \hat{G}_m(x_i)]}{\sum_{i=1}^{n} w_i}$$

  - compute

$$\alpha_m = \log\left(\frac{1 - err_m}{err_m}\right)$$

- set $w_i \leftarrow w_i \, e^{\alpha_m \mathbb{1}[y_i \neq \hat{G}_m(x_i)]}$ $\quad i = 1, \ldots, n$

- The boosted model is

$$\hat{G}(x) = \text{sign}\left(\sum_{m=1}^{M} \alpha_m \hat{G}_m(x) \right)$$

Note

In the first step, we can weight data by resampling each data point with prob $w_i / \sum_{i=1}^{n} w_i$

Ex $n=5$ $\quad w_i = \frac{1}{5}$ @step 1:

- Fit $G_1(x)$ and suppose $y_1 + y_2$ misclassified

  y_3, y_4, y_5 are correctly classified

- $\text{err}_1 = \frac{\frac{1}{5} + \frac{1}{5}}{1} = \frac{2}{5}$

- $\alpha_1 = \log(3/2)$
- update weights $\quad w_3 = w_4 = w_5 = \frac{1}{5}$

$$w_1 = w_2 = \frac{1}{5} e^{\log(3/2)} = \frac{3}{10} \quad \left(> \frac{2}{10} = \frac{1}{5} \right)$$

- Resample + fit $\hat{G}_2(x)$, iterate

Key Boosting works well even when G_i are small models, e.g. a tree stump (tree with one split).

Boosted regression trees

Y = continuous, basic algorithm

- Set $\hat{f}(x) = 0$ + $r_i = y_i$ $\quad i = 1, \ldots, n$

- For each $b = 1, \ldots, B$
  - fit tree $\hat{f}^b$ with d splits ($d+1$ terminal nodes) to data (X, r), $r = (r_1, \ldots, r_n)$
    - X: design matrix
  - Update
    $$\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^b(x)$$
  - Update
    $$r_i = r_i - \lambda \hat{f}^b(x)$$
- Boosted model is
  $$\hat{f}(x) = \sum_{b=1}^{B} \lambda \hat{f}^b(x).$$

Ex $b=1$ $\hat{f}=0$ $r_i=y_i$

- Fit $\hat{h}$ to $(x_1,r_1),\ldots,(x_n,r_n)$

- $\hat{f}=0+\lambda\hat{h}$

- $r_i=r_i-\lambda\hat{h}(x_i)$

$b=2$ - Fit model to $(x_1,r_1-\lambda\hat{h}(x_1)),\ldots,(x_n,r_n-\lambda\hat{h}(x_n))$ and call it $\hat{h}$

- $\hat{f}=[0+\lambda\hat{h}]+\lambda\hat{h}=2\lambda\hat{h}+\lambda\hat{h}$

- $r_i=[r_i-\lambda\hat{h}(x_i)]-\lambda\hat{h}(x_i)$

$=r_i-2\lambda\hat{h}(x_i)$

$\Rightarrow$ after B steps $\hat{f}=\sum_{b=1}^{B}\lambda\hat{h}^b(x)$

Remark - Usually use $d=1$ or 2 (or maybe 3)

$d=$ "interaction depth"

- # of trees B: If too big, then this overfits data.

  B is chosen by C.V.

- Shrinkage param $\lambda > 0$. If $\lambda=1$, fitting + removing a tree at each step which yields a tree.

  λ small ≈ 0.01 or 0.001

Remark Instead of "fitting the data hard", boosting learns the model slowly ($\lambda \approx 0$).