Chapter 9

Bagging, Boosting and Random Forests

9.1 Bagging

Bagging is an acronym for *bootstrap aggregating.* As before, we start with a learning set $\mathcal{L} = \{(X_i, Y_i) : i = 1, 2, \ldots, n\}$. The bagging procedure starts by drawing B bootstrap samples from $\mathcal{L}$; samples obtained *with* replacement. Denote the bootstrap samples by:

$$\mathcal{L}^{*b} = \{(X_i^{*b}, Y_i^{*b}) : i = 1, \ldots, n\} \qquad b = 1, 2, \ldots, B.$$

9.1.1 Bagging Tree-Based Classifiers

Now consider classification, where $Y_i \in \{1, \ldots, K\}$ is a class label attached to X_i. We grow a classification tree $\mathcal{T}^{*b}$ from the bth bootstrap sample $\mathcal{L}^{*b}$. To reduce bias, we grow the tree large without pruning. Suppose (X, Y) is independently drawn from the same distribution, then drop X down each of the bootstrap trees and take the class that appears most often. This classification procedure is known as the *majority-vote rule.*

Number of different observations used in a bootstrap sample The probability that a given observation is not used in a bootstrap sample is $\left(1 - \frac{1}{n}\right)^n \simeq e^{-1} \sim 0.368$, so approximately 36.8% of observations are unused in a bootstrap sample. Let $\mathcal{L} \backslash \mathcal{L}^{*b}$ denote observations of $\mathcal{L}$ not used in bootstrap sample b. These are known as the OOB (out of bag) observations. The OOB observations will function as an independent test set.

For each OOB observation for bootstrap sample b, drop X_i dnown the classification tree $\mathcal{L}^{*b}$. Suppose there are n_i trees for which X_i is OOB. Drop X_i down each of these trees and, for the K classes, compute $\widehat{p}_k(x_i) : i = 1, \ldots, K$ the proportions of trees which classify x_i as class k, $k = 1, \ldots, K$. Then

$$C_{\text{bag}}(x_i) = \arg\max_k \widehat{p}_k(x_i).$$

If y_i is the class variable for observation (x_i, y_i), then the misclassification rate is

$$PE_{\text{bag}} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}(C_{\text{bag}}(x_i) \neq y_i).$$

9.1.2 Bagging Regression-Tree Predictors

For regression, $Y_i \in \mathbb{R}$. From the bth bootstrap sample $\mathcal{L}^{*b}$ we grow a regression tree $\mathcal{T}^{*b}$ and obtain the predictor $\widehat{\mu}^{*b}(x)$. We drop x down each of the B regression trees and then average the predictions

$$\widehat{\mu}_{\text{bag}}(x) = \frac{1}{B}\sum_{b=1}^{B} \widehat{\mu}^{*b}(x).$$

This is the bagged estimate of Y.

To evaluate the predictive abilities of a bagged regression estimate, we again use the OOB approach. Let $(x_i, y_i) \in \mathcal{L}$. We drop x_i down each of the n_i bootstrap trees for which x_i is OOB. The OOB regression estimate $\widehat{\mu}_{\text{bag}}(x_i)$ is found by averaging the n_i bootstrap predicted values.

That is:

$$\widehat{\mu}_{\text{bag}}(x_i) = \frac{1}{n_i}\sum_{b\in\mathcal{N}_i} \widehat{\mu}^{*b}(x_i)$$

where $\mathcal{N}_i$ is the set of n_i bootstrap samples that do not contain (x_i, y_i). We then estimate the generalised error of the bagged estimate by the OOB error rate:

$$\text{PE}_{\text{bag}} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \widehat{\mu}_{\text{bag}}(x_i))^2$$

which is computed as the mean squared error between the bagged estimates and their true response values.

9.2 Boosting

The aim of boosting is to enhance the accuracy of a weak binary classification learning algorithm. It derives from ‘Probably Approximately Correct’ learning from machine learning.

We define a *weak* (or *base*) classifier to be one that correctly classifies slightly more than 50% of the time (i.e. a little better than random guessing). Suppose that we have M base classifiers $C_1, \ldots, C_M$ (where $C_j(x)$ is the class that classifier j assigns to a value x). For an observation $X = x$, the *boosted classifier* is given by:

$$C_\alpha(x) = \text{sign}(f_\alpha(x))$$

where

$$f_\alpha(x) = \sum_{j=1}^{M} \left(\frac{\alpha}{\sum_{j'} \alpha_{j'}} \right) C_j(x)$$

and $\alpha = (\alpha_1, \ldots, \alpha_M)'$ is an M-vector of constant coefficients.

Example We consider junk email. Suppose we have various classifiers to decide whether or not an email is junk. Let e denote a particular email that has just arrived and suppose we have various classifiers based on a single word: 'money', 'free', 'order' 'credit' respectively classify email as junk. Let 1 denote junk and -1 denote serious email.

$$\begin{aligned} C_1(e) &= \begin{cases} 1 & \text{contains word 'money'} \\ -1 & \text{otherwise} \end{cases} \\ C_2(e) &= \begin{cases} 1 & \text{contains word 'free'} \\ -1 & \text{otherwise} \end{cases} \\ C_3(e) &= \begin{cases} 1 & \text{contains word 'order'} \\ -1 & \text{otherwise} \end{cases} \\ C_4(e) &= \begin{cases} 1 & \text{contains word 'credit'} \\ -1 & \text{otherwise} \end{cases} \end{aligned}$$

We may combine these classifiers with non-negative weights summing to 1. For example, suppose we give weights 0.2, 0.1, 0.4, 0.3 respectively to the 4 classifiers, then set

$$f(e) = 0.2C_1(e) + 0.1C_2(e) + 0.4C_3(e) + 0.3C_4(e).$$

The classification we give to an email e is $\text{sign} f(e)$. If it contains the words 'money', 'order' and 'credit', we give it a score $f(e) = 0.2 - 0.1 + 0.4 + 0.3 = 0.8$ and assign it to the class 'spam'.

ADABOOST - Adaptive Boosting The ADABOOST algorithm for binary classification is:

- Input $\mathcal{L} = \{(x_i, y_i) : i = 1, \ldots, n\}$, $y_i \in \{-1, +1\}$ $\mathcal{C} = \{C_1, \ldots, C_M\}$ (M weak classifiers). T number of iterations.
- Initialise weight vector $w_i - (w_{11}, \ldots, w_{n1})'$ where $w_{i1} = \frac{1}{n}$.
- For $t = 1, \ldots, T$
  - Select a weak classifier $C_{j_t}(x) \in \{-1, 1\}$ from $\mathcal{C}$, $j_t \in \{1, \ldots, M\}$ and train it on learning set $\mathcal{L}$ where the ith observation (x_i, y_i) has (normalised) weight w_{it}, $i = 1, \ldots, n$.
  - Compute the weighted prediction error

$$\text{PE}_t = \text{PE}(w_t) = \mathbb{E}_w \left[\mathbf{1}(y_i \neq C_{j_t}(x_i)) \right] = \left(\frac{w_t'}{\mathbf{1}_n' w_t} \right) e_t$$

where $\mathbb{E}_w$ denotes expertation with respect to the probability distribution w_t and e_t is an n-vector with ith entry $(e_t)_i = \mathbf{1}(Y_i \neq C_{j_t}(x_i))$.

- Set $\beta_t = \frac{1}{2}\log\left(\frac{1-PE_t}{PE_t}\right)$
- Update weights

$$w_{i,t+1} = \frac{w_{i,t}}{W_t}\exp\{2\beta_t\mathbf{1}(Y_i \neq C_{j_t}(X_i)) \qquad i = 1,\dots,n$$

where W_t is the normalising constant to make w_{t+1} a probability distribution.

- Output: sign$f(x)$ where $f(x) = \sum_{t=1}^{T}\beta_t C_{j_t}(x) = \sum_{j=1}^{M}\alpha_j C_j(x)$, $\alpha_j = \sum_{t=1}^{T}\beta_t\mathbf{1}(j_t = j)$.

9.3 Random Forests

Random forests start in the same way as bagging starts, with B bootstraps drawn from a learning set $\mathcal{L}$. The difference is in how trees are grown from those samples. A randomisation component is introduced so that, for the tree $\mathcal{T}^{*b}$, each node is split in a random manner. The algorithm is as follows:

- Input $\mathcal{L} = \{(x_i, y_i), i = 1,\dots,n\}$, $y_i \in \{1,\dots,K\}$ (K possible classes), m number of variables to be chosen at each node (where $m << r$) and B number of bootstrap samples.
- For $b = 1,\dots,B$
  - Draw a bootstrap sample $\mathcal{L}^{*b}$ from $\mathcal{L}$.
  - From $\mathcal{L}^{*b}$, grow a tree classifier $\mathcal{T}^{*b}$ using random input selection: at each node, randomly select a subset m of the r variables and, using only the m randomly chosen variables, determine the best split at that node, using either entropy or the Gini index. To reduce bias, grow the tree to a maximum depth with no pruning.
  - The tree $\mathcal{T}^{*b}$ generates an associated random vector θ_b which is independent of the previous $\theta_1,\dots,\theta_{b-1}$ and whose form and dimensionality are determined by the context.
  - Using θ_b and an inpute vector x, define a classifier $h(x,\theta_b)$ having a single vote for the class of x.
- The B randomised tree-structured classifiers $\{h(x,\theta_b)\}$ are collectively called a *random forest.*
- Te observation x is assigned to the majority vote-getting class as determined by the random forest.

The random vector θ_b basically assigns a class designation (most probable class) to each leaf node of $\mathcal{T}^{*b}$.

Let

$$m_b(x, y) = \frac{1}{B}\sum_{b=1}^{B} \mathbf{1}(h(x, \theta_b) = y) - \max_{k \neq y}\left\{\frac{1}{B}\sum_{b=1}^{B} \mathbf{1}(h(x, \theta_b) = k)\right\}$$

be the classification margin and define the generalisation error for the random forest with B trees as:

$$\text{PE}_B = \mathbb{P}_{x,y}(m_B(x, y) < 0)$$

If $m_B(x, y) > 0$, then the correct classifiaction is chosen.

m_B is an approximation to m defined by:

$$m(x, y) = \mathbb{P}_\Theta\left\{h(x, \Theta) = Y\right\} - \max_{k \neq Y} \mathbb{P}(h(x, \Theta) = k),$$

which is the *margin function* of the random forest. This is the amount by which the average number of votes at (x, y) for the correct class exceeds the average vote for any other class.

9.3.1 Bounds on the Generalisation Error

Let $\mu = \mathbb{E}_{X,Y}[m(X, Y)]$ be the expected 'strength' of the classifiers (assumed to be positive). By *strength* we mean a measure of accuracy of a tree in the forest. For the binary case,

$$m(x, y) = 2\mathbb{P}_\Theta(h(x, \Theta) = Y) - 1$$

and $\mu > 0 \Rightarrow \mathbb{E}_{X,Y}[\mathbb{P}_\Theta(h(X, \Theta) = Y) > 0.5$. We compute an upper bound for the generalisation error:

$$\text{PE}^* = \mathbb{P}_{X,Y}\left\{|m(X, Y) - \mathbb{E}_{X,Y}[m(X, Y)]| > \mu\right\}$$

of a random forest. Chebyshev's inequality gives:

$$\text{PE}^* \leq \frac{1}{\mu^2}\text{Var}_{X,Y}(m(X, Y))$$

Let

$$\widetilde{k} = \text{arg max}_{k \neq Y}\mathbb{P}_\Theta(h(X, \Theta) = k)$$

Then

$$m(X, Y) = \mathbb{P}_\Theta(h(X, \Theta) = Y) - \mathbb{P}_\Theta(h(X, \Theta) = \widetilde{k}) = \mathbb{E}_\Theta(m^*(X, Y, \Theta))$$

where

$$m^*(X, Y, \theta) = \mathbf{1}(h(X, \theta) = Y) - \mathbf{1}(h(X, \theta) = \widetilde{k})$$

Then

$$m(X,Y)^2 = \mathbb{E}_{\Theta,\Theta'}[m^*(X,Y,\Theta)m^*(X,Y,\Theta')] = \mathbb{E}_{\Theta,\Theta'}[\rho(\Theta,\Theta')\sigma(\Theta)\sigma(\Theta')]$$

where, for fixed θ and θ',

$$\rho(\theta,\theta') = \mathrm{Corr}_{X,Y}(m^*(X,Y,\theta), m^*(X,Y,\theta'))$$

is the correlation between the raw margin functions of two different members in the forest and

$$\sigma^2(\theta) = \mathrm{Var}_{X,Y}(m^*(X,Y,\theta)).$$

Θ and Θ' are independent. Then

$$\mathrm{Var}_{X,Y}(m(X,Y)) = \overline{\rho}\mathbb{E}_\Theta[\sigma(\Theta)]^2 \le \overline{\rho}\mathbb{E}_\Theta[\sigma^2(\Theta)]$$

where

$$\overline{\rho} = \frac{\mathbb{E}_{\Theta,\Theta'}[\rho(\Theta,\Theta')\sigma(\Theta)\sigma(\Theta')]}{\mathbb{E}_{\Theta,\Theta'}[\sigma(\Theta)\sigma(\Theta')]} = \frac{\mathrm{Var}_{X,Y}(m(X,Y))}{\mathbb{E}_\Theta[\sigma(\Theta)]^2}.$$

With some more work, using $\mathbb{E}_\Theta[\sigma^2(\Theta)] \le 1-\mu^2$, we can obtain

$$\mathrm{PE}^* \le \frac{\overline{\rho}(1-\mu^2)}{\mu^2}.$$