PH 240C Supervised Learning (5): Bagging, random forest, and boosting

Jingshen Wang

October 6, 2021

Bagging (Breiman, 1984), random forest (Breiman, 2001), and boosting (Bartlett et al., 1998) can be broadly categorized as "ensemble learning", which refers to the learning a weighted combination of base models of the form:

$$f(x, \pi) = \sum_{b \in \mathcal{M}} w_b f_b(x),$$

where w_b are tuning parameters (weights), $f_b(x)$ is a given classifier that predict the outcome based on attribute x, and $\mathcal{M}$ represents an index for a class of classifiers. Ensemble learning is sometimes called a committee method, since each base model f_b gets a weighted "vote." The question now is how can we find these weights w_b and the classifiers $f_b(\cdot|\cdot)$. Bagging, random forest, and boosting refers to three different approaches to find the weights and those classifiers.

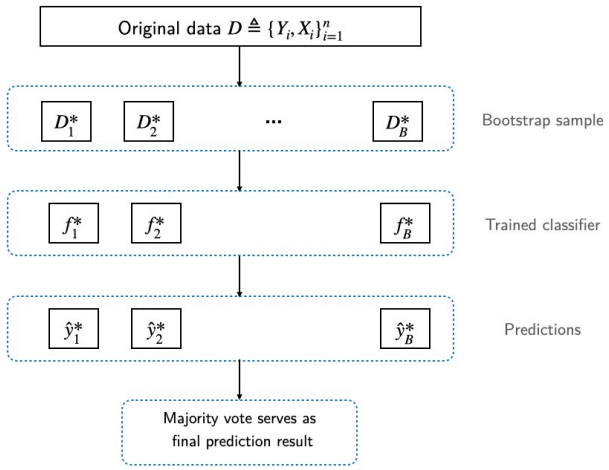

Figure 1: Illustration of Bagging (Breiman, 1984).

1 Bagging

Bagging is also referred to as bootstrap aggregation illustrated in Figure 1. Suppose we have access to a dataset with observations $\mathcal{D} \triangleq \{(Y_i, X_i)\}_{i=1}^n$, and they form an empirical distribution $\widehat{F}_n$. The algorithm works as follows:

1. Draw bootstrap samples from $\mathcal{D}$ with replacement, denoted as $\mathcal{D}_1^*, \ldots, \mathcal{D}_B^*$;
2. On each bootstrap sample, we build a classifier $f_b^*(x)$ (can be tree, or any of your favourite methods);
3. Then final bagged classifier is then

$$f(x) = \frac{1}{B}\sum_{b=1}^{B} f_b^*(x).$$

Now let's look into some details of bagging:

- We can clearly see that bagging sets equal weights to different classifier learnt in different bootstrap samples, i.e., $w_1 = \ldots = w_B = 1/B$. This is actually somewhat suboptimal, as we are not efficiently using all the available information in our original data. Note that in each bootstrap sample $\mathcal{D}_b^*$, not all original data points in $\{(Y_i, X_i)\}_{i=1}^n$ are used in building the classifier $f_b^*(\cdot)$ (Efron and Tibshirani, 1997). We can calculate the probability that a given data point is not drawn in a bootstrap sample as

$$\mathbb{P}(\text{a data point is not chosen}) = \left(1 - \frac{1}{n}\right)^n \to \frac{1}{e} \approx 0.368.$$

  How can we leverage all these unused data points to improve our classifier? The answer of this question is heuristically related to the core idea of Boosting.

- The advantages of bagging is that it is easy to implement, and it reduces variances – so has strong beneficial effect on high variance classifiers. To see the effect of variance reduction, note that bagged estimator has two layers of randomness: one comes from the original data, and another one comes from the bootstrap resampling as we take random samples with replacement from the original data. In fact, each bootstrap sample $\mathcal{D}_b^*$ can be written as

$$\mathcal{D}^* = \{(Y_i \mathrm{v}_i, X_i \mathrm{v}_i)\}_{i=1}^n, \quad \mathrm{v} = (\mathrm{v}_1, \ldots, \mathrm{v}_n) \sim \texttt{Multinomial}(\mathbf{1}_n).$$

  When $B = {}^nP_n$ goes through all possible combination of sample with repeated data points, the bagged classifier can be written as

$$f(x) = \mathbb{E}\big[f_b^*(x)|\mathcal{D}\big],$$

  which indicates $\mathbb{V}\mathrm{ar}[f(x)] \leq \mathbb{V}\mathrm{ar}[f_b^*(x)]$ (how?).

- Now we agree that bagging reduces the variance of a single bootstrapped classifier $f_b^*(\cdot)$, but can we do even better in variance reduction? The variance of $f(x)$ is of the form

$$\mathbb{V}\mathrm{ar}\big[f(x)\big] = \frac{1}{B^2}\mathbb{V}\mathrm{ar}[f_b^*(x)] + \frac{B-1}{B}\mathbb{C}\mathrm{ov}\big[f_{b_1}^*(x), f_{b_2}^*(x)\big].$$

  From this decomposition, we can see that individual tree variance is not the dominant term of the bagged classifier. The covariance between different trees is the main contributing factor for the variance. The intuitive answer of this question leads to the idea of random forest.

- Bagging provides a natural estimate of the test error. We can calculate the "out-of-bag-error" for each learnt classifier on the bootstrap sample. More formally, for each data point $(X_i, Y_i) \in \mathcal{D}$, define

$S_i = \{b : (X_i, Y_i) \notin \mathcal{D}_b\}$ to be the set of all classifiers that are not trained based on the data point (X_i, Y_i). Then the bagged classifier over all these dataset is

$$\widetilde{f}_i(x) = \frac{1}{|S_i|} \sum_{b \in S_i} f_b^*(x).$$

And the out-of-bag-error is simply the average error/loss that all these classifiers yield:

$$\epsilon_{\texttt{OOB}} = \frac{1}{n} \sum_{i=1}^{n} \texttt{loss}\big(\widetilde{f}_i(X_i), Y_i\big).$$

2 Random forest

One of the most famous and useful bagged algorithms is the Random Forest! A Random Forest is essentially nothing else but bagged decision trees, with a slightly modified splitting criteria. The algorithm works as follows:

1. Draw bootstrap samples from $\mathcal{D}$ with replacement, denoted as $\mathcal{D}_1^*, \ldots, \mathcal{D}_B^*$;
2. On each bootstrap sample, further randomly subsample $k \leq d$ attributes (without replacement) and only consider these for your split – this further increases the variance of the trees, but reduces the correlation between different trees. Then, build a classifier $f_b^*(x)$ (can be tree, or any of your favourite methods);
3. Then final random forested classifier is then

$$f(x) = \frac{1}{B} \sum_{b=1}^{B} f_b^*(x).$$

The Random Forest is one of the best, most popular and easiest to use out-of-the-box classifier. There are two reasons for this: (1) The RF only has two hyper-parameters, B and k. It is extremely insensitive to both of these. A good choice for k is $k = \sqrt{d}$ (where d denotes the number of attribute). Ideally, if we can afford the computational cost, we would choose B as large as possible. (2) Decision trees do not require a lot of preprocessing. For example, the features can be of different scale, magnitude, or slope. This can be highly advantageous in scenarios with heterogeneous data, for example the medical settings where features could be things like blood pressure, age, gender, ..., each of which is recorded in completely different units.

While random forests are naturally less interpretable than individual decision trees, where we can trace a decision via a rule sets, it is possible (and common) to compute the so-called "variable importance" of the attributes that means, we can infer how important a feature is for the overall prediction. For classification tree, the variable importance is typically calculated based on Gini-index.

3 Boosting

There are two broad categories of boosting: Adaptive boosting and gradient boosting. Adaptive and gradient boosting rely on the same concept of boosting "weak learners" to "strong learners." See Figure 2 for illustration. Boosting is an iterative process, where the training set is reweighted, at each iteration, based on

Figure 2: Illustration of weak and strong learners.

mistakes a weak leaner made (i.e., misclassifications); the two approaches, adaptive and gradient boosting, differ mainly regarding how the weights are updated and how the classifiers are combined. Since we have not discussed gradient-based optimization, in this lecture, we will focus on adaptive boosting. In particular, we will focus on AdaBoost.

Intuitively, we can outline the general boosting procedure as follows:

1. Initialize a weight vector with uniform weights–each data point plays an equal role in building the first classifier;

2. Train a weighted learner $f_1(x)$ based on the sample with equal weights, and calculate its training classification error, denoted as $\epsilon_1 \in [0, 1]$;

3. Increased the weights for misclassified data points (as the algorithm can improve its accuracy from furthering learning these data points), we define $w_1 = \frac{1}{2} \log \frac{1-\epsilon_1}{\epsilon_1}$ and

$$\mathrm{v}_{i1} = \begin{cases} \exp(-w_1) & \text{if } f_1(X_i) = Y_i \\ \exp(w_1) & \text{if } f_1(X_i) \neq Y_i \end{cases}$$

   then normalize these weights so they sum up to one.

4. Train a weighted learner $f_2(x)$ based on the sample with equal weights, and calculate its weighted training classification error:

$$\epsilon_2 = \sum_{i=1}^{n} \mathrm{v}_{i1} \mathbf{1}(f(X_i) \neq Y_i),$$

   and we update the weights as

$$\mathrm{v}_{i2} = \begin{cases} \mathrm{v}_{i1} \cdot \exp(-w_2) & \text{if } f_2(X_i) = Y_i \\ \mathrm{v}_{i1} \cdot \exp(w_2) & \text{if } f_2(X_i) \neq Y_i \end{cases}, \quad w_2 = \frac{1}{2} \log \frac{1-\epsilon_2}{\epsilon_2}.$$

   This step essentially tells us that we need to decrease the weights for the ith data point whenever it is classified correctly.

5. Repeat the above steps until $\epsilon_b > 1/2$. Then the adaboost classifier is of the from:

$$f(x) = \sum_{b=1}^{B} w_b f_b(x).$$

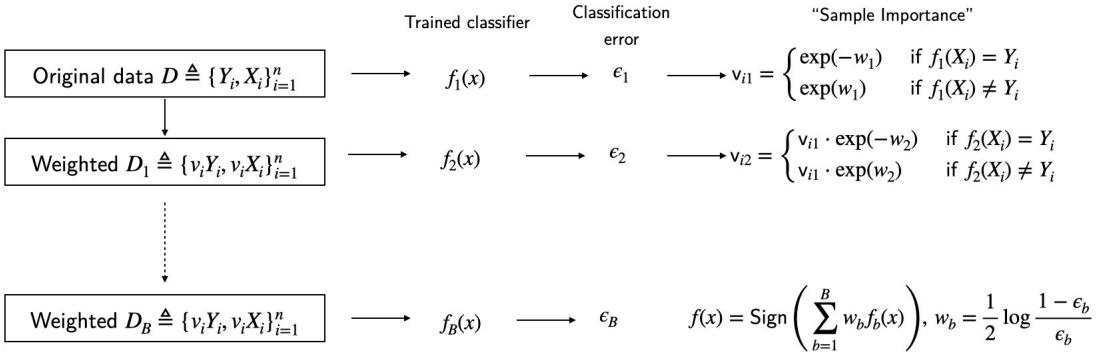

Figure 3: Illustration of AdaBoost.

References

Peter Bartlett, Yoav Freund, Wee Sun Lee, and Robert E Schapire. Boosting the margin: A new explanation for the effectiveness of voting methods. *The annals of statistics*, 26(5):1651–1686, 1998.

Leo Breiman. Classification and regression trees. Technical report, 1984.

Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

Bradley Efron and Robert Tibshirani. Improvements on cross-validation: the 632+ bootstrap method. *Journal of the American Statistical Association*, 92(438):548–560, 1997.