CSCI 5525 Machine Learning Fall 2019

Lecture 18: Bagging and Random Forest

April 2020

Lecturer: Steven Wu *Scribe: Steven Wu*

Revisit bias variance tradeoff

Again, we are given a dataset $D = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, drawn i.i.d. from some distribution $P(X, Y)$. Recall the following definitions from the last lecture:

Expected Label for x.

$$\bar{y}(x) = E_P [Y \mid X = x]$$

Suppose we use some machine learning algorithm $\mathcal{A}$ that takes the data set D as input and outputs a predictor, denoted $f_D = \mathcal{A}(D)$. We can define:

Expected Test Error for f_D.

$$E_{(x,y)\sim P} \left[(f_D(x) - y)^2 \right].$$

Now we should also take into account that the data set D is also drawn randomly from P, and so there is randomness in the predictor produced by $\mathcal{A}$ as well.

Expected Classifier for a given algorithm $\mathcal{A}$.

$$\bar{h} = E_{D\sim P^n} [f_D]$$

We can also use the fact that f_D is a random variable to compute the expected test error only given $\mathcal{A}$, taking the expectation also over D.

Expected Test Error given $\mathcal{A}$.

$$E_{\substack{(x,y)\sim P \\ D\sim P^n}} \left[(f_D(x) - y)^2 \right]$$

Now recall the Bias / Variance decomposition:

$$\underbrace{\mathbb{E}[(f_D(x) - y)^2]}_{\text{Error}} = \underbrace{\mathbb{E}[(f_D(x) - \bar{f}(x))^2]}_{\text{Variance}} + \underbrace{\mathbb{E}[(\bar{f}(x) - \bar{y}(x))^2]}_{\text{Bias}} + \underbrace{\mathbb{E}[(\bar{y}(x) - y)^2]}_{\text{Noise}}$$

Now we will introduce a method for reducing the variance term.

Bagging (Bootstrap Aggregating) [Breiman 96]

To reduce the variance, we want $f_D \to \bar{f}$. Ideally, we want to use the law of large numbers and obtain m independent training sets $D_1, D_2, ..., D_m$ drawn from P^n. Train a classifier on each one and average result:

$$\hat{f} = \frac{1}{m} \sum_{i=1}^{m} f_{D_i} \to \bar{f}$$

This average of multiple classifiers is another ensemble of classifiers. When m is sufficiently large, we can show that $\hat{f} \to \bar{f}$, and so the variance also vanishes:

$$\mathbb{E}[(\hat{f}(x) - \bar{f}(x))^2] \to 0$$

However, we don't have m data sets $D_1, ..., D_m$, we only have the one training dataset D.

Bagging The idea is to simulate drawing from P by drawing from the training set D: sample m data sets $D_1, \ldots, D_m$ from D with replacement, such that $|D_j| = |D|$. For each D_j train a classifier $h_j()$ The final predictor is given by the *bagged predictor* :

$$h(\mathbf{x}) = \frac{1}{m} \sum_{j=1}^{m} h_j(\mathbf{x})$$

Advantages of Bagging

- It is easy to implement, and it can work with any models.
- It reduces variance, so it is often used along with predictors with high variance, e.g., large decision trees.
- Since bagging produces many predictors, you obtain a mean and variance among the predictions given by the predictors. One can use the variance term to quantify uncertainty.
- Bagging provides the so-called out-of-bag error, which is an unbiased estimate of the test error. Observe that each training point $(x_i, y_i) \in D$ was not chosen for certain sets D_j, (x_i, y_i) can be viewed as a test example for the corresponding predictors h_j.

  More formally, for each training point $(x_i, y_i) \in D$ let $S_i = \{j | (\mathbf{x}_i, y_i) \notin D_j\}$, that is S_i is a set of all the training sets D_k that do not contain (x_i, y_i). Consider the following average predictor:

  $$\tilde{h}_i(\mathbf{x}) = \frac{1}{|S_i|} \sum_{k \in S_i} h_k(\mathbf{x}).$$

  The out-of-bag error is then defined as:

  $$\text{Err}_{\text{OOB}} = \frac{1}{n} \sum_{(x_i, y_i) \in D} \ell(y_i, \tilde{h}_i(x_i)).$$

  Hence, bagging provides an estimate for the test error without using a test set!

Random Forest

Random forest is an algorithm we obtain by applying bagging to decision trees:

- Sample m data sets $D_1, \ldots, D_m$ from D with replacement.
- For each D_j train a full decision tree $h_j()$ with one small modification: before each split randomly subsample $k \leq d$ features and only consider these for your split.
- The final classifier is $h(\mathbf{x}) = \frac{1}{m} \sum_{j=1}^{m} h_j(\mathbf{x})$.

Subsampling features. The subsampling of the features for each split helps *decorrelate* the set of decision trees. This can be useful because intuitively we want the decision trees to make mistakes over different regions and so they can complement each other when we the average. The size of the feature set for each split is a hyperparameter. The author of random forest made the following suggestions on k. For classification, the default value for k is $\sqrt{d}$ and the minimum node size is one. For regression, the default value for k is $d/3$.

Sample splitting. Here is a variant of random forest: for each $j \in [m]$, split each training set D_j into two partitions $D_j = D_j^1 \cup D_j^2$. Build the tree on D_j^1 and estimate the leaf labels on D_j^2. During the process of tree building, the algorithm ensures that each leaf has only a single point in D_j^2 in it. This helps reduces biases of the each decision tree.